

ASN1C

ASN.1 Compiler
Version 7.2
PER Runtime
Reference Manual

The software described in this document is furnished under a license agreement and may be used only in accordance with the terms of this agreement.

Copyright Notice

Copyright ©1997–2018 Objective Systems, Inc. All rights reserved.

This document may be distributed in any form, electronic or otherwise, provided that it is distributed in its entirety and that the copyright and this notice are included.

Author's Contact Information

Comments, suggestions, and inquiries regarding ASN1C may be submitted via electronic mail to info@obj-sys.com.

Contents

1	ASN1C PER Runtime Classes and Library Functions	1
2	Module Index	3
2.1	Modules	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Module Documentation	11
6.1	PER C++ Runtime Classes.	11
6.1.1	Detailed Description	11
6.2	PER Message Buffer Classes	12
6.2.1	Detailed Description	12
6.3	PER Runtime Library Functions.	13
6.3.1	Detailed Description	18
6.3.2	Macro Definition Documentation	18
6.3.2.1	pd_bit	18
6.3.2.2	PD_BYTE_ALIGN0	18

6.3.2.3	PD_CHECKSEQOFLN	19
6.3.2.4	pd_DateTime	19
6.3.2.5	pd_NumericString	19
6.3.2.6	pe_DateTime	20
6.3.2.7	pe_NumericString	20
6.3.2.8	PU_GETPADBITS	20
6.3.2.9	PU_GETSIZECONSTRAINT	20
6.3.2.10	PU_SETCHARSET	21
6.3.2.11	PU_SETCTXTBITOFFSET	21
6.3.2.12	PU_SETSIZECONSTRAINT	21
6.4	PER C Decode Functions.	22
6.4.1	Detailed Description	24
6.4.2	Macro Definition Documentation	24
6.4.2.1	pd_moveBitCursor	24
6.4.3	Function Documentation	24
6.4.3.1	pd_16BitConstrainedString()	24
6.4.3.2	pd_32BitConstrainedString()	25
6.4.3.3	pd_BigInteger()	26
6.4.3.4	pd_BigIntegerEx()	26
6.4.3.5	pd_BigIntegerValue()	27
6.4.3.6	pd_BitString()	27
6.4.3.7	pd_BitString32()	28
6.4.3.8	pd_BitString64()	28
6.4.3.9	pd_BMPString()	29
6.4.3.10	pd_byte_align()	29
6.4.3.11	pd_ChoiceOpenTypeExt()	30
6.4.3.12	pd_ConsInt16()	30
6.4.3.13	pd_ConsInt64()	30

6.4.3.14	<code>pd_ConstInt8()</code>	31
6.4.3.15	<code>pd_ConstInteger()</code>	32
6.4.3.16	<code>pd_ConstrainedString()</code>	32
6.4.3.17	<code>pd_ConstrainedStringEx()</code>	33
6.4.3.18	<code>pd_ConstrFixedLenStringEx()</code>	33
6.4.3.19	<code>pd_ConstUInt16()</code>	34
6.4.3.20	<code>pd_ConstUInt16SignedBound()</code>	35
6.4.3.21	<code>pd_ConstUInt64()</code>	35
6.4.3.22	<code>pd_ConstUInt64SignedBound()</code>	36
6.4.3.23	<code>pd_ConstUInt8()</code>	36
6.4.3.24	<code>pd_ConstUInt8SignedBound()</code>	37
6.4.3.25	<code>pd_ConstUnsigned()</code>	37
6.4.3.26	<code>pd_ConstUnsignedSignedBound()</code>	38
6.4.3.27	<code>pd_ConstWholeNumber()</code>	38
6.4.3.28	<code>pd_ConstWholeNumber64()</code>	39
6.4.3.29	<code>pd_DateStr()</code>	39
6.4.3.30	<code>pd_DateTimeStr()</code>	40
6.4.3.31	<code>pd_Duration()</code>	40
6.4.3.32	<code>pd_DynBitString()</code>	41
6.4.3.33	<code>pd_DynBitString64()</code>	41
6.4.3.34	<code>pd_DynOctetString()</code>	42
6.4.3.35	<code>pd_DynOctetString64()</code>	42
6.4.3.36	<code>pd_GetBinStrDataOffset()</code>	42
6.4.3.37	<code>pd_GetComponentLength()</code>	43
6.4.3.38	<code>pd_GetComponentLength64()</code>	43
6.4.3.39	<code>pd_Interval()</code>	44
6.4.3.40	<code>pd_isFragmented()</code>	45
6.4.3.41	<code>pd_Length()</code>	45

6.4.3.42	<code>pd_Length64()</code>	45
6.4.3.43	<code>pd_ObjectIdentifier()</code>	46
6.4.3.44	<code>pd_OctetString()</code>	46
6.4.3.45	<code>pd_OctetString64()</code>	47
6.4.3.46	<code>pd_oid64()</code>	48
6.4.3.47	<code>pd_OpenType()</code>	48
6.4.3.48	<code>pd_OpenTypeExt()</code>	49
6.4.3.49	<code>pd_Real()</code>	49
6.4.3.50	<code>pd_Real10()</code>	50
6.4.3.51	<code>pd_RelativeOID()</code>	50
6.4.3.52	<code>pd_SemiConsInt16()</code>	51
6.4.3.53	<code>pd_SemiConsInt64()</code>	51
6.4.3.54	<code>pd_SemiConsInt8()</code>	52
6.4.3.55	<code>pd_SemiConsInteger()</code>	52
6.4.3.56	<code>pd_SemiConsUInt16()</code>	53
6.4.3.57	<code>pd_SemiConsUInt16SignedBound()</code>	53
6.4.3.58	<code>pd_SemiConsUInt64()</code>	54
6.4.3.59	<code>pd_SemiConsUInt64SignedBound()</code>	54
6.4.3.60	<code>pd_SemiConsUInt8()</code>	55
6.4.3.61	<code>pd_SemiConsUInt8SignedBound()</code>	55
6.4.3.62	<code>pd_SemiConsUnsigned()</code>	56
6.4.3.63	<code>pd_SemiConsUnsignedSignedBound()</code>	56
6.4.3.64	<code>pd_SmallLength()</code>	57
6.4.3.65	<code>pd_SmallNonNegWholeNumber()</code>	57
6.4.3.66	<code>pd_TimeStr()</code>	58
6.4.3.67	<code>pd_UnconsInt16()</code>	58
6.4.3.68	<code>pd_UnconsInt64()</code>	59
6.4.3.69	<code>pd_UnconsInt8()</code>	59

6.4.3.70	<code>pd_UnconsInteger()</code>	60
6.4.3.71	<code>pd_UnconsLength()</code>	60
6.4.3.72	<code>pd_UnconsLength64()</code>	61
6.4.3.73	<code>pd_UnconsUInt16()</code>	61
6.4.3.74	<code>pd_UnconsUInt64()</code>	62
6.4.3.75	<code>pd_UnconsUInt8()</code>	62
6.4.3.76	<code>pd_UnconsUnsigned()</code>	63
6.4.3.77	<code>pd_UniversalString()</code>	63
6.4.3.78	<code>pd_VarWidthCharString()</code>	64
6.4.3.79	<code>pd_YearInt()</code>	64
6.4.3.80	<code>uperDecConstrFixedLenString()</code>	65
6.4.3.81	<code>uperDecConstrString()</code>	65
6.5	PER C Encode Functions	67
6.5.1	Detailed Description	68
6.5.2	Macro Definition Documentation	68
6.5.2.1	<code>pe_bit</code>	69
6.5.2.2	<code>pe_bits</code>	69
6.5.2.3	<code>pe_CheckBuffer</code>	69
6.5.2.4	<code>pe_ConsInteger</code>	70
6.5.2.5	<code>pe_ConsUnsigned</code>	70
6.5.2.6	<code>pe_ConsUnsignedSignedBound</code>	71
6.5.2.7	<code>pe_octets</code>	72
6.5.2.8	<code>pe_SemiConsInteger</code>	72
6.5.2.9	<code>pe_SemiConsUnsigned</code>	73
6.5.3	Function Documentation	73
6.5.3.1	<code>pe_16BitConstrainedString()</code>	73
6.5.3.2	<code>pe_2sCompBinInt()</code>	74
6.5.3.3	<code>pe_2sCompBinInt64()</code>	74

6.5.3.4	pe_32BitConstrainedString()	75
6.5.3.5	pe_aligned_octets()	75
6.5.3.6	pe_BigInteger()	77
6.5.3.7	pe_bits64()	77
6.5.3.8	pe_BitString()	78
6.5.3.9	pe_BitStringExt()	78
6.5.3.10	pe_BMPString()	79
6.5.3.11	pe_byte_align()	80
6.5.3.12	pe_ChoiceTypeExt()	80
6.5.3.13	pe_ConsInt64()	81
6.5.3.14	pe_ConstrainedString()	81
6.5.3.15	pe_ConstrainedStringEx()	82
6.5.3.16	pe_ConsUInt64()	82
6.5.3.17	pe_ConsUInt64SignedBound()	83
6.5.3.18	pe_ConsWholeNumber()	84
6.5.3.19	pe_ConsWholeNumber64()	84
6.5.3.20	pe_DateStr()	85
6.5.3.21	pe_DateTimeStr()	85
6.5.3.22	pe_Duration()	86
6.5.3.23	pe_GetIntLen()	86
6.5.3.24	pe_GetMsgBitCnt()	86
6.5.3.25	pe_GetMsgPtr()	87
6.5.3.26	pe_GetMsgPtr64()	87
6.5.3.27	pe_GetMsgPtrU()	88
6.5.3.28	pe_Interval()	88
6.5.3.29	pe_Length()	88
6.5.3.30	pe_NonNegBinInt()	89
6.5.3.31	pe_NonNegBinInt64()	89

6.5.3.32	pe_ObjectIdentifier()	90
6.5.3.33	pe_OctetString()	90
6.5.3.34	pe_oid64()	91
6.5.3.35	pe_OpenType()	91
6.5.3.36	pe_OpenTypeEnd()	92
6.5.3.37	pe_OpenTypeExt()	92
6.5.3.38	pe_OpenTypeExtBits()	93
6.5.3.39	pe_OpenTypeStart()	93
6.5.3.40	pe_Real()	95
6.5.3.41	pe_Real10()	95
6.5.3.42	pe_RelativeOID()	96
6.5.3.43	pe_SemiConsInt64()	96
6.5.3.44	pe_SemiConsUInt64()	97
6.5.3.45	pe_SemiConsUInt64SignedBound()	97
6.5.3.46	pe_SemiConsUnsignedSignedBound()	98
6.5.3.47	pe_SmallLength()	98
6.5.3.48	pe_SmallNonNegWholeNumber()	99
6.5.3.49	pe_TimeStr()	99
6.5.3.50	pe_UnconsInt64()	100
6.5.3.51	pe_UnconsInteger()	100
6.5.3.52	pe_UnconsLength()	101
6.5.3.53	pe_UnconsUInt64()	101
6.5.3.54	pe_UnconsUnsigned()	102
6.5.3.55	pe_UniversalString()	102
6.5.3.56	pe_VarWidthCharString()	103
6.5.3.57	pe_YearInt()	103
6.5.3.58	uperEncConstrString()	104

6.6	PER C Utility Functions	105
-----	-------------------------	-----

6.6.1	Detailed Description	106
6.6.2	Function Documentation	106
6.6.2.1	pe_resetBuffer()	106
6.6.2.2	pu_addSizeConstraint()	106
6.6.2.3	pu_bindump()	107
6.6.2.4	pu_checkSizeExt()	107
6.6.2.5	pu_freeContext()	108
6.6.2.6	pu_GetLibInfo()	108
6.6.2.7	pu_GetLibVersion()	108
6.6.2.8	pu_getMsgLen()	109
6.6.2.9	pu_getMsgLenBits()	109
6.6.2.10	pu_hexdump()	110
6.6.2.11	pu_initContext()	110
6.6.2.12	pu_initContextBuffer()	111
6.6.2.13	pu_initFieldList()	111
6.6.2.14	pu_initRtxDiagBitFieldList()	111
6.6.2.15	pu_insLenField()	112
6.6.2.16	pu_isFixedSize()	112
6.6.2.17	pu_newContext()	112
6.6.2.18	pu_newField()	113
6.6.2.19	pu_popName()	113
6.6.2.20	pu_pushElemName()	114
6.6.2.21	pu_pushName()	114
6.6.2.22	pu_set16BitCharSet()	114
6.6.2.23	pu_set16BitCharSetFromRange()	115
6.6.2.24	pu_set32BitCharSet()	115
6.6.2.25	pu_set32BitCharSetFromRange()	116
6.6.2.26	pu_setBuffer()	116
6.6.2.27	pu_setCharSet()	117

7	Class Documentation	119
7.1	ASN1PERDecodeBuffer Class Reference	119
7.1.1	Detailed Description	120
7.1.2	Constructor & Destructor Documentation	120
7.1.2.1	ASN1PERDecodeBuffer() [1/5]	120
7.1.2.2	ASN1PERDecodeBuffer() [2/5]	120
7.1.2.3	ASN1PERDecodeBuffer() [3/5]	121
7.1.2.4	ASN1PERDecodeBuffer() [4/5]	121
7.1.2.5	ASN1PERDecodeBuffer() [5/5]	121
7.1.3	Member Function Documentation	122
7.1.3.1	byteAlign()	122
7.1.3.2	isA()	122
7.1.3.3	peekByte()	122
7.1.3.4	readBinaryFile()	123
7.1.3.5	readBytes()	123
7.2	ASN1PEREncodeBuffer Class Reference	124
7.2.1	Detailed Description	124
7.2.2	Constructor & Destructor Documentation	125
7.2.2.1	ASN1PEREncodeBuffer() [1/4]	125
7.2.2.2	ASN1PEREncodeBuffer() [2/4]	125
7.2.2.3	ASN1PEREncodeBuffer() [3/4]	125
7.2.2.4	ASN1PEREncodeBuffer() [4/4]	126
7.2.3	Member Function Documentation	126
7.2.3.1	byteAlign()	126
7.2.3.2	encodeBit()	126
7.2.3.3	encodeBits()	127
7.2.3.4	getMsgBitCnt()	127
7.2.3.5	getMsgCopy()	128

7.2.3.6	getMsgPtr()	128
7.2.3.7	init()	128
7.2.3.8	isA()	128
7.3	ASN1PERMessageBuffer Class Reference	129
7.3.1	Detailed Description	130
7.3.2	Constructor & Destructor Documentation	130
7.3.2.1	ASN1PERMessageBuffer() [1/4]	130
7.3.2.2	ASN1PERMessageBuffer() [2/4]	130
7.3.2.3	ASN1PERMessageBuffer() [3/4]	131
7.3.2.4	ASN1PERMessageBuffer() [4/4]	131
7.3.3	Member Function Documentation	131
7.3.3.1	binDump()	132
7.3.3.2	getMsgLen()	132
7.3.3.3	hexDump()	132
7.3.3.4	isAligned()	132
7.3.3.5	setBuffer()	133
7.3.3.6	setTrace()	133
7.4	BinDumpBuffer Struct Reference	134
7.5	PERField Struct Reference	134
8	File Documentation	135
8.1	asn1per.h File Reference	135
8.1.1	Detailed Description	144
8.2	asn1PerCppTypes.h File Reference	144
8.2.1	Detailed Description	145
Index		147

Chapter 1

ASN1C PER Runtime Classes and Library Functions

The **ASN.1 C++ runtime classes** are wrapper classes that provide an object-oriented interface to the ASN.1 C runtime library functions. The classes described in this manual are derived from the common classes documented in the ASN1C C/C++ Common runtime manual. They are specific to the Packed Encoding Rules (PER) as defined in the X.691 ITU-T standard. These PER specific C++ runtime classes include the PER message buffer classes.

The **ASN.1 PER Runtime Library** contains the low-level constants, types, and functions that are assembled by the compiler to encode/decode more complex structures.

This library consists of the following items:

- A global include file ("asn1per.h") that is compiled into all generated source files.
- An object library of functions that are linked with the C functions after compilation with a C compiler.

In general, programmers will not need to be too concerned with the details of these functions. The ASN.1 compiler generates calls to them in the C or C++ source files that it creates. However, the functions in the library may also be called on their own in applications requiring their specific functionality.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

PER C++ Runtime Classes.	11
PER Message Buffer Classes	12
PER Runtime Library Functions.	13
PER C Decode Functions.	22
PER C Encode Functions.	67
PER C Utility Functions	105

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ASN1MessageBuffer	
ASN1PERMessageBuffer	129
ASN1PERDecodeBuffer	119
ASN1PEREncodeBuffer	124
BinDumpBuffer	134
PERField	134

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- [ASN1PERDecodeBuffer](#) 119
- [ASN1PEREncodeBuffer](#) 124
- [ASN1PERMessageBuffer](#) 129
- [BinDumpBuffer](#) 134
- [PERField](#) 134

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

asn1per.h	135
asn1PerCppType.h	144

Chapter 6

Module Documentation

6.1 PER C++ Runtime Classes.

Modules

- [PER Message Buffer Classes](#)

6.1.1 Detailed Description

6.2 PER Message Buffer Classes

Classes

- class [ASN1PERMessageBuffer](#)
- class [ASN1PEREncodeBuffer](#)
- class [ASN1PERDecodeBuffer](#)

6.2.1 Detailed Description

The ASN.1 C++ runtime classes are wrapper classes that provide an object-oriented interface to the ASN.1 C runtime library functions. These classes are derived from the common classes documented in the ASN1C C/C++ Common Runtime Functions manual and are specific to the Packed Encoding Rules (PER). These classes manage the buffers for encoding and decoding ASN.1 PER messages.

6.3 PER Runtime Library Functions.

Modules

- [PER C Decode Functions.](#)
- [PER C Encode Functions.](#)
- [PER C Utility Functions](#)

Classes

- struct [PERField](#)
- struct [BinDumpBuffer](#)

Macros

- `#define ASN_K_EXTENUM 999`
- `#define MAX_BIGINTBYTES (1024)`
- `#define OSYEAR_BASIC OSUINTCONST(0x8000000)`
- `#define OSYEAR_PROLEPTIC OSUINTCONST(0x4000000)`
- `#define OSYEAR_NEGATIVE OSUINTCONST(0x2000000)`
- `#define OSYEAR_L(n) ((OSUINT32)(n) << 28)`
- `#define OSYEAR_MASK (OSYEAR_BASIC|OSYEAR_PROLEPTIC|OSYEAR_NEGATIVE|OSYEAR_L(0xF))`
- `#define OSANY (OSYEAR_NEGATIVE|OSYEAR_L(5))`
- `#define OSANY_MASK (OSYEAR_NEGATIVE|OSYEAR_L(0xF))`
- `#define OSCENTURY 0x4000u`
- `#define OSYEAR 0x2000u`
- `#define OSMONTH 0x1000u`
- `#define OSWEEK 0x0800u`
- `#define OSDAY 0x0400u`
- `#define OSHOURS 0x0200u`
- `#define OSMINUTES 0x0100u`
- `#define OSSECONDS 0x0080u`
- `#define OSUTC 0x0040u`
- `#define OSDIFF 0x0020u`
- `#define OSFRACTION 0x000Fu`
- `#define OSDURATION 0x0010u`
- `#define PU_SETCHARSET(csetvar, canset, abits, ubits)`
- `#define PU_INSLENFLD(pctxt)`
- `#define PU_NEWFIELD(pctxt, suffix)`
- `#define PU_PUSHNAME(pctxt, name)`
- `#define PU_PUSHELEMNAME(pctxt, idx)`
- `#define PU_POPNAME(pctxt)`
- `#define PU_SETBITOFFSET(pctxt)`
- `#define PU_SETBITCOUNT(pctxt)`
- `#define PU_SETOPENTYPEFLDLIST(pMainBFList, pOpenTypeBFList)`
- `#define EXTPERMETHOD`
- `#define EXTPERCLASS`
- `#define PD_BIT(pctxt, pvalue) DEC_BIT(pctxt,pvalue)`

- #define **PU_SETSIZECONSTRAINT**(pctxt, rootLower, rootUpper, extLower, extUpper)
- #define **PU_INITSIZECONSTRAINT**(pctxt) PU_SETSIZECONSTRAINT(pctxt,0,0,0,0)
- #define **PU_GETSIZECONSTRAINT**(pctxt, extbit)
- #define **PU_GETCTXTBITOFFSET**(pctxt) (((pctxt)->buffer.byteIndex * 8) + (8 - (pctxt)->buffer.bitOffset))
- #define **PU_GETPADBITS**(pctxt) (((pctxt)->buffer.bitOffset == 8) ? 0 : (pctxt)->buffer.bitOffset)
- #define **PU_SETCTXTBITOFFSET**(pctxt, _bitOffset)
- #define **PD_BYTE_ALIGN0**(pctxt)
- #define **PD_BYTE_ALIGN** PD_BYTE_ALIGN0
- #define **PD_CHECKSEQOFLEN**(pctxt, numElements, minElemBits)
- #define **pd_bit**(pctxt, pvalue) rtxDecBit(pctxt,pvalue)
- #define **pd_bits**(pctxt, pvalue, nbits) rtxDecBits(pctxt,pvalue,nbits)
- #define **pd_octets**(pctxt, pBuffer, bufsiz, nbits) rtxDecBitsToByteArray(pctxt,pBuffer,bufsiz,nbits)
- #define **pe_GeneralString**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_GraphicString**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_T61String**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_TeletexString**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_VideotexString**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_ObjectDescriptor**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_UTF8String**(pctxt, value, permCharSet) [pe_VarWidthCharString](#)(pctxt, value)
- #define **pe_IA5String**(pctxt, value, permCharSet) [pe_ConstrainedStringEx](#) (pctxt, value, permCharSet, 8, 7, 7)
- #define **pe_NumericString**(pctxt, value, permCharSet)
- #define **pe_PrintableString**(pctxt, value, permCharSet) [pe_ConstrainedStringEx](#) (pctxt, value, permCharSet, 8, 7, 7)
- #define **pe_VisibleString**(pctxt, value, permCharSet) [pe_ConstrainedStringEx](#) (pctxt, value, permCharSet, 8, 7, 7)
- #define **pe_ISO646String** pe_IA5String
- #define **pe_GeneralizedTime** pe_IA5String
- #define **pe_UTCTime** pe_GeneralizedTime
- #define **pd_GeneralString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_GraphicString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_VideotexString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_TeletexString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_T61String**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_ObjectDescriptor**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_UTF8String**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_IA5String**(pctxt, pvalue, permCharSet) [pd_ConstrainedStringEx](#) (pctxt, pvalue, permCharSet, 8, 7, 7)
- #define **pd_NumericString**(pctxt, pvalue, permCharSet)
- #define **pd_PrintableString**(pctxt, pvalue, permCharSet) [pd_ConstrainedStringEx](#) (pctxt, pvalue, permCharSet, 8, 7, 7)
- #define **pd_VisibleString**(pctxt, pvalue, permCharSet) [pd_ConstrainedStringEx](#) (pctxt, pvalue, permCharSet, 8, 7, 7)
- #define **pd_ISO646String** pd_IA5String
- #define **pd_GeneralizedTime** pd_IA5String
- #define **pd_UTCTime** pd_GeneralizedTime
- #define **pe_GetMsgLen** [pu_getMsgLen](#)
- #define **pe_ExpandBuffer**(pctxt, nbytes) rtxExpandOutputBuffer(pctxt,nbytes)
- #define **pd_AnyCentury**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSCENTURY)
- #define **pd_AnyCenturyInt**(pctxt, pvalue) [pd_UnconsInteger](#) (pctxt, pvalue)
- #define **pd_AnyDate**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pd_AnyYear**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR)

- #define **pd_AnyYearInt**(pctxt, pvalue) **pd_UnconsInteger** (pctxt, pvalue)
- #define **pd_AnyYearDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSANY|OSYEAR|OSDAY)
- #define **pd_AnyYearMonth**(pctxt, string) **pd_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH)
- #define **pd_AnyYearMonthDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pd_AnyYearWeek**(pctxt, string) **pd_DateStr** (pctxt, string, OSANY|OSYEAR|OSWEEK)
- #define **pd_AnyYearWeekDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSANY|OSYEAR|OSWEEK|OSDAY)
- #define **pd_Century**(pctxt, string) **pd_DateStr** (pctxt, string, OSCENTURY)
- #define **pd_CenturyInt**(pctxt, pvalue) **pd_ConsUInt8** (pctxt, pvalue, 0, 99)
- #define **pd_Date**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR_BASIC|OSYEAR|OSMONTH|OSDAY);
- #define **pd_DateTime**(pctxt, string)
- #define **pd_DurationInterval**(pctxt, string) **pd_Duration** (pctxt, string, FALSE)
- #define **pd_DurationEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pd_DurationEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pd_DurationEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, OSDURATI↵
ON, flags)
- #define **pd_Hours**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS)
- #define **pd_HoursUtc**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSUTC)
- #define **pd_HoursAndDiff**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSDIFF)
- #define **pd_HoursAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|(n))
- #define **pd_HoursUtcAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSUTC|(n))
- #define **pd_HoursAndDiffAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSDIFF|(n))
- #define **pd_Minutes**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES)
- #define **pd_MinutesUtc**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSUTC)
- #define **pd_MinutesAndDiff**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSDIFF)
- #define **pd_MinutesAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|(n))
- #define **pd_MinutesUtcAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OS↵
UTC|(n))
- #define **pd_MinutesAndDiffAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUT↵
ES|OSDIFF|(n))
- #define **pd_RecStartEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pd_RecStartEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pd_RecStartEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pd_RecDurationInterval**(pctxt, string) **pd_Duration** (pctxt, string, TRUE)
- #define **pd_RecStartDateDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, OSDU↵
RATION)
- #define **pd_RecStartTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, OSDU↵
RATION)
- #define **pd_RecStartDateTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, O↵
SDURATION)
- #define **pd_RecDurationEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, OSDURATION, flags)
- #define **pd_RecDurationEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, OSDURATION, flags)
- #define **pd_RecDurationEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, OSDUR↵
ATION, flags)
- #define **pd_StartEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pd_StartEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pd_StartEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pd_StartDateDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, OSDURA↵
TION)

- #define **pd_StartTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, OSDURATION)
- #define **pd_StartDateDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, OSDURATION)
- #define **pd_TimeOfDay**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS)
- #define **pd_TimeOfDayUtc**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSUTC)
- #define **pd_TimeOfDayAndDiff**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSDIFF)
- #define **pd_TimeOfDayAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|(n))
- #define **pd_TimeOfDayUtcAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSUTC|(n))
- #define **pd_TimeOfDayAndDiffAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSDIFF|(n))
- #define **pd_Year**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR)
- #define **pd_YearDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSDAY)
- #define **pd_YearMonth**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSMONTH)
- #define **pd_YearMonthDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSMONTH|OSDAY);
- #define **pd_YearWeek**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSWEEK)
- #define **pd_YearWeekDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSWEEK|OSDAY)
- #define **pe_AnyCentury**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSCENTURY)
- #define **pe_AnyCenturyInt**(pctxt, value) **pe_UnconsInteger** (pctxt, value)
- #define **pe_AnyDate**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pe_AnyYear**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR)
- #define **pe_AnyYearInt**(pctxt, value) **pe_UnconsInteger** (pctxt, value)
- #define **pe_AnyYearDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSDAY)
- #define **pe_AnyYearMonth**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH)
- #define **pe_AnyYearMonthDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pe_AnyYearWeek**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSWEEK)
- #define **pe_AnyYearWeekDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSWEEK|OSDAY)
- #define **pe_Century**(pctxt, string) **pe_DateStr** (pctxt, string, OSCENTURY)
- #define **pe_CenturyInt**(pctxt, value) **pe_ConsUnsigned** (pctxt, value, 0, 99)
- #define **pe_Date**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR_BASIC|OSYEAR|OSMONTH|OSDAY)
- #define **pe_DateTime**(pctxt, string)
- #define **pe_DurationInterval**(pctxt, string) **pe_Duration** (pctxt, string, FALSE)
- #define **pe_DurationEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pe_DurationEndTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pe_DurationEndDateTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pe_Hours**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS)
- #define **pe_Hours**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS)
- #define **pe_HoursUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSUTC)
- #define **pe_HoursUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSUTC)
- #define **pe_HoursAndDiff**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSDIFF)
- #define **pe_HoursAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|(n))
- #define **pe_HoursUtcAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSUTC|(n))
- #define **pe_HoursAndDiffAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSDIFF|(n))
- #define **pe_Minutes**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES)

- #define **pe_MinutesUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSUTC)
- #define **pe_MinutesAndDiff**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSDIFF)
- #define **pe_MinutesAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|(n))
- #define **pe_MinutesUtcAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OS←
UTC|(n))
- #define **pe_MinutesAndDiffAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUT←
ES|OSDIFF|(n))
- #define **pe_RecStartEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pe_RecStartEndTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pe_RecStartEndDateTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pe_RecDurationInterval**(pctxt, string) **pe_Duration** (pctxt, string, TRUE)
- #define **pe_RecStartDateDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, OSDU←
RATION)
- #define **pe_RecStartTimeDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, OSDU←
RATION)
- #define **pe_RecStartDateTimeDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, O←
SDURATION)
- #define **pe_RecDurationEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, OSDURATION,
flags)
- #define **pe_RecDurationEndTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, OSDURATION,
flags)
- #define **pe_RecDurationEndDateTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, OSDUR←
ATION, flags)
- #define **pe_StartEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pe_StartEndTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pe_StartEndDateTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pe_StartDateDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, flags, OSDURA←
TION)
- #define **pe_StartTimeDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, flags, OSDURA←
TION)
- #define **pe_StartDateTimeDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, flags, OSD←
URATION)
- #define **pe_TimeOfDay**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS)
- #define **pe_TimeOfDayUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|O←
SUTC)
- #define **pe_TimeOfDayAndDiff**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECON←
DS|OSDIFF)
- #define **pe_TimeOfDayAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OS←
SECONDS|(n))
- #define **pe_TimeOfDayUtcAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUT←
ES|OSSECONDS|OSUTC|(n))
- #define **pe_TimeOfDayAndDiffAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINU←
TES|OSSECONDS|OSDIFF|(n))
- #define **pe_Year**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR)
- #define **pe_YearDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR|OSDAY)
- #define **pe_YearMonth**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR|OSMONTH)
- #define **pe_YearMonthDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR|OSMONTH|OSDAY)
- #define **pe_YearWeek**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR|OSWEEK)
- #define **pe_YearWeekDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR|OSWEEK|OSDAY)

Typedefs

- typedef struct [PERField](#) **PERField**

Functions

- int **pu_checkSizeConstraint** (OSCTXT *pctxt, int size)
- Asn1SizeCnst * **pu_getSizeConstraint** (OSCTXT *pctxt, OSBOOL extbit)
- int **pu_getBitOffset** (OSCTXT *pctxt)
- void **pu_setBitOffset** (OSCTXT *pctxt, int bitOffset)

6.3.1 Detailed Description

The ASN.1 Packed Encoding Rules (PER) runtime library contains the low-level constants, types, and functions that are assembled by the compiler to encode/decode more complex structures. The PER low-level C encode/decode functions are identified by their prefixes: pe_ for PER encode, pd_ for PERdecode, and pu_ for PER utility functions.

6.3.2 Macro Definition Documentation

6.3.2.1 pd_bit

```
#define pd_bit(  
    pctxt,  
    pvalue ) rtxDecBit(pctxt,pvalue)
```

perutil

6.3.2.2 PD_BYTE_ALIGN0

```
#define PD_BYTE_ALIGN0(  
    pctxt )
```

Value:

```
((!(pctxt)->buffer.aligned) ? 0 : \  
((pctxt)->buffer.bitOffset != 8) ? ( \  
(pctxt)->buffer.byteIndex++, \  
(pctxt)->buffer.bitOffset = 8, \  
0) : 0 \  
)
```


6.3.2.3 PD_CHECKSEQOFLLEN

```
#define PD_CHECKSEQOFLLEN(  
    pctxt,  
    numElements,  
    minElemBits )
```

Value:

```
((pctxt->buffer.size > 0) ? \  
((numElements * minElemBits) > (pctxt->buffer.size * 8)) ? \  
LOG_RTERR (pctxt,ASN_E_INVLEN) : 0) : 0)
```

6.3.2.4 pd_DateTime

```
#define pd_DateTime(  
    pctxt,  
    string )
```

Value:

```
pd\_DateTimeStr (pctxt, string, \  
OSYEAR_BASIC|OSYEAR|OSMONTH|OSDAY|OSHOURS|OSMINUTES|OSSECONDS);
```

6.3.2.5 pd_NumericString

```
#define pd_NumericString(  
    pctxt,  
    pvalue,  
    permCharSet )
```

Value:

```
pd\_ConstrainedStringEx (pctxt, pvalue, \  
(permCharSet == 0)?NUM_CANSET:permCharSet, 4, 4, 4)
```

6.3.2.6 pe_DateTime

```
#define pe_DateTime(  
    pctxt,  
    string )
```

Value:

```
pe_DateTimeStr (pctxt, string, \  
OSYEAR_BASIC|OSYEAR|OSMONTH|OSDAY|OSHOURS|OSMINUTES|OSSECONDS)
```

6.3.2.7 pe_NumericString

```
#define pe_NumericString(  
    pctxt,  
    value,  
    permCharSet )
```

Value:

```
pe_ConstrainedStringEx (pctxt, value, \  
(permCharSet == 0)?NUM_CANSET:permCharSet, 4, 4, 4)
```

6.3.2.8 PU_GETPADBITS

```
#define PU_GETPADBITS(  
    pctxt ) ((pctxt)->buffer.bitOffset == 8) ? 0 : (pctxt)->buffer.bitOffset)
```

This macro returns the number of padding bits in the last byte following an encode or decode operation.

6.3.2.9 PU_GETSIZECONSTRAINT

```
#define PU_GETSIZECONSTRAINT(  
    pctxt,  
    extbit )
```

Value:

```
((extbit) ? \  
&ACINFO(pctxt)->sizeConstraint.ext : &ACINFO(pctxt)->sizeConstraint.root)
```

6.3.2.10 PU_SETCHARSET

```
#define PU_SETCHARSET(  
    csetvar,  
    canset,  
    abits,  
    ubits )
```

Value:

```
csetvar.charSet.nchars = 0; \  
csetvar.canonicalSet = canset; \  
csetvar.canonicalSetSize = sizeof(canset)-1; \  
csetvar.canonicalSetBits = pu_bitcnt(csetvar.canonicalSetSize); \  
csetvar.charSetUnalignedBits = ubits; \  
csetvar.charSetAlignedBits = abits;
```

6.3.2.11 PU_SETCTXTBITOFFSET

```
#define PU_SETCTXTBITOFFSET(  
    pctxt,  
    _bitOffset )
```

Value:

```
do { \  
(pctxt)->buffer.byteIndex = (_bitOffset / 8); \  
(pctxt)->buffer.bitOffset = (OSUINT16)(8 - (_bitOffset % 8)); \  
} while(0)
```

6.3.2.12 PU_SETSIZECONSTRAINT

```
#define PU_SETSIZECONSTRAINT(  
    pctxt,  
    rootLower,  
    rootUpper,  
    extLower,  
    extUpper )
```

Value:

```
ACINFO(pctxt)->sizeConstraint.root.lower = rootLower; \  
ACINFO(pctxt)->sizeConstraint.root.upper = rootUpper; \  
ACINFO(pctxt)->sizeConstraint.ext.lower = extLower; \  
ACINFO(pctxt)->sizeConstraint.ext.upper = extUpper
```

6.4 PER C Decode Functions.

Macros

- #define `pd_moveBitCursor`(pctx, bitOffset) `rtxMoveBitCursor`(pctx,bitOffset)

Functions

- int `pd_BigInteger` (OSCTXT *pctx, const char **ppvalue)
- int `pd_BigIntegerEx` (OSCTXT *pctx, const char **ppvalue, int radix)
- int `pd_BigIntegerValue` (OSCTXT *pctx, const char **ppvalue, int radix, OSUINT32 nbytes)
- int `pd_BitString` (OSCTXT *pctx, OSUINT32 *numbits_p, OSOCTET *buffer, OSSIZE bufsiz)
- int `pd_BitString64` (OSCTXT *pctx, OSSIZE *numbits_p, OSOCTET *buffer, OSSIZE bufsiz)
- int `pd_BitString32` (OSCTXT *pctx, ASN1BitStr32 *pbitstr, OSSIZE lower, OSSIZE upper)
- int `pd_BMPString` (OSCTXT *pctx, ASN1BMPString *pvalue, Asn116BitCharSet *permCharSet)
- int `pd_UniversalString` (OSCTXT *pctx, ASN1UniversalString *pvalue, Asn132BitCharSet *permCharSet)
- int `pd_byte_align` (OSCTXT *pctx)
- int `pd_ChoiceOpenTypeExt` (OSCTXT *pctx, const OSOCTET **object_p2, OSSIZE *pnumocts)
- int `pd_ConstInteger` (OSCTXT *pctx, OSINT32 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstInt8` (OSCTXT *pctx, OSINT8 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstInt16` (OSCTXT *pctx, OSINT16 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstInt64` (OSCTXT *pctx, OSINT64 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstUnsigned` (OSCTXT *pctx, OSUINT32 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConstUnsignedSignedBound` (OSCTXT *pctx, OSUINT32 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstUInt8` (OSCTXT *pctx, OSUINT8 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConstUInt8SignedBound` (OSCTXT *pctx, OSUINT8 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstUInt16` (OSCTXT *pctx, OSUINT16 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConstUInt16SignedBound` (OSCTXT *pctx, OSUINT16 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstUInt64` (OSCTXT *pctx, OSUINT64 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConstUInt64SignedBound` (OSCTXT *pctx, OSUINT64 *pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConstWholeNumber` (OSCTXT *pctx, OSUINT32 *padjusted_value, OSUINT32 range_value)
- int `pd_ConstWholeNumber64` (OSCTXT *pctx, OSUINT64 *padjusted_value, OSUINT64 range_value)
- int `pd_ConstrainedString` (OSCTXT *pctx, const char **string, Asn1CharSet *pCharSet)
- int `pd_ConstrainedStringEx` (OSCTXT *pctx, const char **string, const char *charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int `pd_ConstrFixedLenStringEx` (OSCTXT *pctx, char *strbuf, size_t bufsiz, const char *charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int `pd_16BitConstrainedString` (OSCTXT *pctx, Asn116BitCharString *pString, Asn116BitCharSet *pCharSet)
- int `pd_32BitConstrainedString` (OSCTXT *pctx, Asn132BitCharString *pString, Asn132BitCharSet *pCharSet)
- int `pd_DateStr` (OSCTXT *pctx, const char **string, OSUINT32 flags)
- int `pd_DateTimeStr` (OSCTXT *pctx, const char **string, OSUINT32 flags)
- int `pd_Duration` (OSCTXT *pctx, const char **string, OSBOOL rec)
- int `pd_DynBitString` (OSCTXT *pctx, ASN1DynBitStr *pBitStr)
- int `pd_DynBitString64` (OSCTXT *pctx, ASN1DynBitStr64 *pBitStr)
- int `pd_DynOctetString` (OSCTXT *pctx, ASN1DynOctStr *pOctStr)
- int `pd_DynOctetString64` (OSCTXT *pctx, OSDynOctStr64 *pOctStr)
- int `pd_GetBinStrDataOffset` (OSCTXT *pctx, OSUINT32 *pnumbits, OSBOOL bitStrFlag)
- int `pd_GetComponentLength` (OSCTXT *pctx, OSUINT32 itemBits)
- int `pd_GetComponentLength64` (OSCTXT *pctx, OSUINT32 itemBits, OSSIZE *plength)

- int [pd_Interval](#) (OSCTXT *pctxt, const char **string, OSBOOL rec, OSUINT32 startFlags, OSUINT32 endFlags)
- int [pd_Length](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- int [pd_Length64](#) (OSCTXT *pctxt, OSSIZE *pvalue)
- int [pd_ObjectIdentifier](#) (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int [pd_oid64](#) (OSCTXT *pctxt, ASN1OID64 *pvalue)
- int [pd_RelativeOID](#) (OSCTXT *pctxt, ASN1OBJID *pvalue)
- int [pd_OctetString](#) (OSCTXT *pctxt, OSUINT32 *pnumocts, OSOCTET *buffer, OSUINT32 bufsiz)
- int [pd_OctetString64](#) (OSCTXT *pctxt, OSSIZE *pnumocts, OSOCTET *buffer, OSSIZE bufsiz)
- int [pd_OpenType](#) (OSCTXT *pctxt, const OSOCTET **object_p2, OSSIZE *pnumocts)
- int [pd_OpenTypeExt](#) (OSCTXT *pctxt, const OSOCTET **object_p2, OSSIZE *pnumocts)
- int [pd_Real](#) (OSCTXT *pctxt, OSREAL *pvalue)
- int [pd_SmallLength](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- int [pd_SmallNonNegWholeNumber](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- int [pd_SemiConsInteger](#) (OSCTXT *pctxt, OSINT32 *pvalue, OSINT64 lower)
- int [pd_SemiConsUnsigned](#) (OSCTXT *pctxt, OSUINT32 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUnsignedSignedBound](#) (OSCTXT *pctxt, OSINT32 *pvalue, OSINT64 lower)
- int [pd_SemiConsInt8](#) (OSCTXT *pctxt, OSINT8 *pvalue, OSINT64 lower)
- int [pd_SemiConsUInt8](#) (OSCTXT *pctxt, OSUINT8 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUInt8SignedBound](#) (OSCTXT *pctxt, OSUINT8 *pvalue, OSINT64 lower)
- int [pd_SemiConsInt16](#) (OSCTXT *pctxt, OSINT16 *pvalue, OSINT64 lower)
- int [pd_SemiConsUInt16](#) (OSCTXT *pctxt, OSUINT16 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUInt16SignedBound](#) (OSCTXT *pctxt, OSUINT16 *pvalue, OSINT64 lower)
- int [pd_SemiConsInt64](#) (OSCTXT *pctxt, OSINT64 *pvalue, OSINT64 lower)
- int [pd_SemiConsUInt64](#) (OSCTXT *pctxt, OSUINT64 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUInt64SignedBound](#) (OSCTXT *pctxt, OSUINT64 *pvalue, OSINT64 lower)
- int [pd_TimeStr](#) (OSCTXT *pctxt, const char **string, OSUINT32 flags)
- int [pd_UnconsInteger](#) (OSCTXT *pctxt, OSINT32 *pvalue)
- int [pd_UnconsLength](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- EXTPERMETHOD int [pd_UnconsLength64](#) (OSCTXT *pctxt, OSSIZE *pvalue)
- int [pd_UnconsUnsigned](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- int [pd_UnconsInt8](#) (OSCTXT *pctxt, OSINT8 *pvalue)
- int [pd_UnconsUInt8](#) (OSCTXT *pctxt, OSUINT8 *pvalue)
- int [pd_UnconsInt16](#) (OSCTXT *pctxt, OSINT16 *pvalue)
- int [pd_UnconsUInt16](#) (OSCTXT *pctxt, OSUINT16 *pvalue)
- int [pd_UnconsInt64](#) (OSCTXT *pctxt, OSINT64 *pvalue)
- int [pd_UnconsUInt64](#) (OSCTXT *pctxt, OSUINT64 *pvalue)
- int [pd_VarWidthCharString](#) (OSCTXT *pctxt, const char **pvalue)
- int [pd_YearInt](#) (OSCTXT *pctxt, OSINT32 *pvalue)
- int [pd_Real10](#) (OSCTXT *pctxt, const char **ppvalue)
- OSBOOL [pd_isFragmented](#) (OSCTXT *pctxt)
- void [pd_OpenTypeStart](#) (OSCTXT *pctxt, OSSIZE *pSavedSize, OSINT16 *pSavedBitOff)
- int [pd_OpenTypeEnd](#) (OSCTXT *pctxt, OSSIZE savedSize, OSINT16 savedBitOff)
- int [uperDecConstrString](#) (OSCTXT *pctxt, const char **string, const char *charSet, OSUINT32 nbits, OSUINT32 canSetBits)
- int [uperDecConstrFixedLenString](#) (OSCTXT *pctxt, char *strbuf, size_t bufsiz, const char *charSet, OSUINT32 nbits, OSUINT32 canSetBits)

6.4.1 Detailed Description

PER runtime library decode functions handle the decoding of the primitive ASN.1 data types and length variables. Calls to these functions are assembled in the C source code generated by the ASN1C compiler to decode complex ASN.1 structures. These functions are also directly callable from within a user's application program if the need to decode a primitive item exists.

The procedure to decode a primitive data item is as follows:

1. Call the `pu_newContext` or `pu_initContext` function to specify the address of the buffer containing the encoded ASN.1 data to be decoded and whether the data is aligned, or unaligned.
2. Call the specific decode function to decode the value.

6.4.2 Macro Definition Documentation

6.4.2.1 `pd_moveBitCursor`

```
#define pd_moveBitCursor(  
    pctxt,  
    bitOffset ) rtxMoveBitCursor(pctxt,bitOffset)
```

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>bitOffset</i>	The bit offset inside the message buffer.

6.4.3 Function Documentation

6.4.3.1 `pd_16BitConstrainedString()`

```
int pd_16BitConstrainedString (  
    OSCTXT * pctxt,  
    Asn116BitCharString * pString,  
    Asn116BitCharSet * pCharSet )
```

This function will encode a constrained ASN.1 character string. This function is normally not called directly but rather is called from Useful Type Character String encode functions that deal with 16-bit strings. The only function that does not release is the `pe_BMPString` function.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pString</i>	Character string to be encoded. The structure includes a count field containing the number of characters to encode and an array of unsigned short integers to hold the 16-bit characters to be encoded.
<i>pCharSet</i>	Pointer to the constraining character set. This contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.2 pd_32BitConstrainedString()

```
int pd_32BitConstrainedString (
    OSCTXT * pctxt,
    Asn132BitCharString * pString,
    Asn132BitCharSet * pCharSet )
```

This function will encode a constrained ASN.1 character string. This function is normally not called directly but rather is called from Useful Type Character String encode functions that deal with 32-bit strings. The only function that does not release is the pe_UniversalString function.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pString</i>	Character string to be encoded. The structure includes a count field containing the number of characters to encode and an array of unsigned short integers to hold the 32-bit characters to be encoded.
<i>pCharSet</i>	Pointer to the constraining character set. This contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.3 pd_BigInteger()

```
int pd_BigInteger (
    OSCTXT * pctxt,
    const char ** ppvalue )
```

This function decodes a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes. These variables are stored in character string constant variables. They are represented as hexadecimal strings starting with "0x" prefix. If it is necessary to convert a hexadecimal string to another radix, then use the ::rtxBigIntSetStr / ::rtxBigIntToString functions.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ppvalue</i>	Pointer to a character pointer variable to receive the decoded unsigned value. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function. The decoded variable is represented as a decimal string starting with no prefix.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.4 pd_BigIntegerEx()

```
int pd_BigIntegerEx (
    OSCTXT * pctxt,
    const char ** ppvalue,
    int radix )
```

This variant of the pd_BigInteger function allows the user to select the radix of the decoded integer string.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ppvalue</i>	Pointer to a character pointer variable to receive the decoded unsigned value. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function. The decoded variable is represented as a decimal string starting with no prefix.
<i>radix</i>	Radix to be used for decoded string. Valid values are 2, 8, 10, or 16.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.5 pd_BigIntegerValue()

```
int pd_BigIntegerValue (
    OSCTXT * pctxt,
    const char ** ppvalue,
    int radix,
    OSUINT32 nbytes )
```

This function decodes only the value portion of an integer field. It is assume the length was decode separately.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>ppvalue</i>	Pointer to a character pointer variable to receive the decoded unsigned value. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function. The decoded variable is represented as a decimal string starting with no prefix.
<i>radix</i>	Radix to be used for decoded string. Valid values are 2, 8, 10, or 16.
<i>nbytes</i>	Length in bytes of the value component.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.6 pd_BitString()

```
int pd_BitString (
    OSCTXT * pctxt,
    OSUINT32 * numbits_p,
    OSOCTET * buffer,
    OSSIZE bufsiz )
```

This function will decode a value of the ASN.1 bit string type whose maximum size is is known in advance. The ASN1C compiler generates a call to this function to decode bit string productions or elements that contain a size constraint.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>numbits</i> ↔ <i>_p</i>	Pointer to an unsigned integer variable to receive decoded number of bits.
<i>buffer</i>	Pointer to a fixed-size or pre-allocated array of <i>bufsiz</i> octets to receive a decoded bit string.
<i>bufsiz</i>	Length (in octets) of the buffer to receive the decoded bit string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.7 pd_BitString32()

```
int pd_BitString32 (
    OSCTXT * pctxt,
    ASN1BitStr32 * pbitstr,
    OSSIZE lower,
    OSSIZE upper )
```

This version of `pd_BitString` will decode a bit string into the standard 32-bit bit string type which can hold up to 32 bits.

See also

[pd_BitString](#)

6.4.3.8 pd_BitString64()

```
int pd_BitString64 (
    OSCTXT * pctxt,
    OSSIZE * numbits_p,
    OSOCTET * buffer,
    OSSIZE bufsiz )
```

64-bit version of `pd_BitString`. Length is returned in a size-typed variable which scales to 64-bits on 64-bit machines.

See also

[pd_BitString](#)

6.4.3.9 pd_BMPString()

```
int pd_BMPString (
    OSCTXT * pctxt,
    ASN1BMPString * pvalue,
    Asn116BitCharSet * permCharSet )
```

This function will decode a variable of the ASN.1 BMP character string. This differs from the decode routines for the character strings previously described in that the BMP string type is based on 16-bit characters. A 16-bit character string is modeled using an array of unsigned short integers.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to character string structure to receive the decoded result. The structure includes a count field containing the number of characters and an array of unsigned short integers to hold the 16-bit character values.
<i>permCharSet</i>	A pointer to the constraining character set. This contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.10 pd_byte_align()

```
int pd_byte_align (
    OSCTXT * pctxt )
```

This function will position the decode bit cursor on the next byte boundary.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.11 pd_ChoiceOpenTypeExt()

```
int pd_ChoiceOpenTypeExt (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSSIZE * pnumocts )
```

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p2</i>	A pointer to an open type variable to receive the decoded data.
<i>pnumocts</i>	A pointer to an unsigned buffer of bufsiz octets to receive decoded data.

6.4.3.12 pd_ConstInt16()

```
int pd_ConstInt16 (
    OSCTXT * pctxt,
    OSINT16 * pvalue,
    OSINT64 lower,
    OSINT64 upper )
```

This function will decode a 16-bit integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 16-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.13 pd_ConstInt64()

```
int pd_ConstInt64 (
    OSCTXT * pctxt,
```

```

    OSINT64 * pvalue,
    OSINT64 lower,
    OSINT64 upper )

```

This function will decode a 64-bit integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as 64-bit integer.
<i>upper</i>	Upper range value, represented as 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.14 pd_ConsInt8()

```

int pd_ConsInt8 (
    OSCTXT * pctxt,
    OSINT8 * pvalue,
    OSINT64 lower,
    OSINT64 upper )

```

This function will decode an 8-bit integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 8-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.15 pd_ConstInteger()

```
int pd_ConstInteger (
    OSCTXT * pctxt,
    OSINT32 * pvalue,
    OSINT64 lower,
    OSINT64 upper )
```

This function will decode an integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.16 pd_ConstrainedString()

```
int pd_ConstrainedString (
    OSCTXT * pctxt,
    const char ** string,
    Asn1CharSet * pCharSet )
```

This function decodes a constrained string value. This is a deprecated version of the function provided for backward compatibility.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to const char* to receive decoded string. Memory will be allocated for this variable using internal memory management functions.
<i>pCharSet</i>	Pointer to a character set descriptor structure. This contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.17 pd_ConstrainedStringEx()

```
int pd_ConstrainedStringEx (
    OSCTXT * pctxt,
    const char ** string,
    const char * charSet,
    OSUINT32 abits,
    OSUINT32 ubits,
    OSUINT32 canSetBits )
```

This function decodes a constrained string value. This version of the function allows all of the required permitted alphabet constraint parameters to be passed in as arguments.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to const char* to receive decoded string. Memory will be allocated for this variable using internal memory management functions.
<i>charSet</i>	String containing permitted alphabet character set. Can be null if no character set was specified.
<i>abits</i>	Number of bits in a character set character (aligned).
<i>ubits</i>	Number of bits in a character set character (unaligned).
<i>canSetBits</i>	Number of bits in a character from the canonical set representing this string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.18 pd_ConstrFixedLenStringEx()

```
int pd_ConstrFixedLenStringEx (
    OSCTXT * pctxt,
    char * strbuf,
    size_t bufsiz,
    const char * charSet,
    OSUINT32 abits,
    OSUINT32 ubits,
    OSUINT32 canSetBits )
```

This function decodes a constrained string value into a fixed-size buffer. This function allows all of the required permitted alphabet constraint parameters to be passed in as arguments.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>strbuf</i>	Pointer to character array to receive decoded string.
<i>bufsiz</i>	Size of strbuf character array.
<i>charSet</i>	String containing permitted alphabet character set. Can be null if no character set was specified.
<i>abits</i>	Number of bits in a character set character (aligned).
<i>ubits</i>	Number of bits in a character set character (unaligned).
<i>canSetBits</i>	Number of bits in a character from the canonical set representing this string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.19 pd_ConsUInt16()

```
int pd_ConsUInt16 (
    OSCTXT * pctxt,
    OSUINT16 * pvalue,
    OSUINT64 lower,
    OSUINT64 upper )
```

This function will decode a 16-bit unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 16-bit unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.20 pd_ConsUInt16SignedBound()

```
int pd_ConsUInt16SignedBound (
    OSCTXT * pctxt,
    OSUINT16 * pvalue,
    OSINT64 lower,
    OSINT64 upper )
```

This function will decode a 16-bit unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 16-bit unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.21 pd_ConsUInt64()

```
int pd_ConsUInt64 (
    OSCTXT * pctxt,
    OSUINT64 * pvalue,
    OSUINT64 lower,
    OSUINT64 upper )
```

This function will decode a 64-bit unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

6.4.3.22 pd_ConsUInt64SignedBound()

```
int pd_ConsUInt64SignedBound (
    OSCTXT * pctxt,
    OSUINT64 * pvalue,
    OSINT64 lower,
    OSINT64 upper )
```

This function will decode a 64-bit unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.23 pd_ConsUInt8()

```
int pd_ConsUInt8 (
    OSCTXT * pctxt,
    OSUINT8 * pvalue,
    OSUINT64 lower,
    OSUINT64 upper )
```

This function will decode an 8-bit unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 8-bit unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.24 pd_ConsUInt8SignedBound()

```
int pd_ConsUInt8SignedBound (
    OSCTXT * pctxt,
    OSUINT8 * pvalue,
    OSINT64 lower,
    OSINT64 upper )
```

This function will decode an 8-bit unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 8-bit unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.25 pd_ConsUnsigned()

```
int pd_ConsUnsigned (
    OSCTXT * pctxt,
    OSUINT32 * pvalue,
    OSUINT64 lower,
    OSUINT64 upper )
```

This function will decode an unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value. 37
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.26 pd_ConsUnsignedSignedBound()

```
int pd_ConsUnsignedSignedBound (
    OSCTXT * pctxt,
    OSUINT32 * pvalue,
    OSINT64 lower,
    OSINT64 upper )
```

This function will decode an unsigned integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.27 pd_ConsWholeNumber()

```
int pd_ConsWholeNumber (
    OSCTXT * pctxt,
    OSUINT32 * padjusted_value,
    OSUINT32 range_value )
```

This function decodes a constrained whole number as specified in Section 10.5 of the X.691 standard.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>padjusted_value</i>	Pointer to unsigned adjusted integer value to receive decoded result. To get the final value, this value is added to the lower boundary of the range.
<i>range_value</i>	Unsigned integer value specifying the total size of the range. This is obtained by subtracting the lower range value from the upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.28 pd_ConsWholeNumber64()

```
int pd_ConsWholeNumber64 (
    OSCTXT * pctxt,
    OSUINT64 * padjusted_value,
    OSUINT64 range_value )
```

This function decodes a constrained whole number as specified in Section 10.5 of the X.691 standard, represented as 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>padjusted_value</i>	Pointer to 64-bit unsigned adjusted integer value to receive decoded result. To get the final value, this value is added to the lower boundary of the range.
<i>range_value</i>	Unsigned 64-bit integer value specifying the total size of the range. This is obtained by subtracting the lower range value from the upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.29 pd_DateStr()

```
int pd_DateStr (
    OSCTXT * pctxt,
    const char ** string,
    OSUINT32 flags )
```

This function will decode an ISO 8601 DATE type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string variable to receive decoded value in string form (YYYY:MM:DD) and ect.
<i>flags</i>	Set of flags: OSANY OSCENTURY OSYEAR OSMONTH OSWEEK OSDAY.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.30 pd_DateTimeStr()

```
int pd_DateTimeStr (
    OSCTXT * pctxt,
    const char ** string,
    OSUINT32 flags )
```

This function will decode an ISO 8601 DATE-TIME type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string variable to receive decoded value in string form (YYYY-MM-DDTHH:MM:SS) and ect.
<i>flags</i>	Set of flags: OSANY OSCENTURY OSYEAR OSMONTH OSWEEK OSDAY OSHOURS OSMINUTES OSSECONDS OSUTC OSDIFF n. n - set digit number of fraction part.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.31 pd_Duration()

```
int pd_Duration (
    OSCTXT * pctxt,
    const char ** string,
    OSBOOL rec )
```

This function will decode an ISO 8601 DURATION types.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string variable to receive decoded value in string form (PnYnMnDTnHnMnS).
<i>rec</i>	Decode recursive interval (Rn/).

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.32 pd_DynBitString()

```
int pd_DynBitString (
    OSCTXT * pctx,
    ASN1DynBitStr * pBitStr )
```

This function will decode a variable of the ASN.1 BIT STRING type. This function allocates dynamic memory to store the decoded result. The ASN1C compiler generates a call to this function to decode an unconstrained bit string production or element.

Parameters

<i>pctx</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pBitStr</i>	Pointer to a dynamic bit string structure to receive the decoded result. This structure contains a field to hold the number of decoded bits and a pointer to an octet string to hold the decoded data. Memory is allocated by the decoder using the <code>rxMemAlloc</code> function. This memory is tracked within the context and released when the <code>pu_freeContext</code> function is invoked.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.33 pd_DynBitString64()

```
int pd_DynBitString64 (
    OSCTXT * pctx,
    ASN1DynBitStr64 * pBitStr )
```

64-bit version of `pd_DynBitString`.

See also

[pd_DynBitString](#)

6.4.3.34 pd_DynOctetString()

```
int pd_DynOctetString (
    OSCTXT * pctxt,
    ASN1DynOctStr * pOctStr )
```

This function will decode a value of the ASN.1 octet string type whose maximum size is known in advance. The ASN1C compiler generates a call to this function to decode octet string productions or elements that contain a size constraint.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pOctStr</i>	A pointer to a dynamic octet string to receive the decoded result.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.35 pd_DynOctetString64()

```
int pd_DynOctetString64 (
    OSCTXT * pctxt,
    OSDynOctStr64 * pOctStr )
```

64-bit version of pd_DynOctetString.

See also

[pd_DynOctetString](#)

6.4.3.36 pd_GetBinStrDataOffset()

```
int pd_GetBinStrDataOffset (
    OSCTXT * pctxt,
    OSUINT32 * pnumbits,
    OSBOOL bitStrFlag )
```

This function gets the offset in bits to the data field within a PER-encoded binary string (i.e a BIT or OCTET STRING).

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pnumbits</i>	A pointer to an unsigned integer to receive the bit count value.
<i>bitStrFlag</i>	TRUE if type being operaton is a BIT STRING; FALSE if OCTET STRING.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.37 pd_GetComponentLength()

```
int pd_GetComponentLength (
    OSCTXT * pctxt,
    OSUINT32 itemBits )
```

This function gets the total length of a PER-encoded component. In the case of a fragmented length, it will look ahead and add up each of the individual length components.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>itemBits</i>	The size of the specific entity.

Returns

Completion status of operation:

- ≥ 0 = success, total parsed length
- negative return value is error.

6.4.3.38 pd_GetComponentLength64()

```
int pd_GetComponentLength64 (
    OSCTXT * pctxt,
    OSUINT32 itemBits,
    OSSIZE * plength )
```

This function gets the total length of a PER-encoded component. In the case of a fragmented length, it will look ahead and add up each of the individual length components. This version will return length value up to 64-bits in size in 64-bit systems.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>itemBits</i>	The size of the specific entity.
<i>plength</i>	Pointer to variable to hold parsed length value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.39 pd_Interval()

```
int pd_Interval (
    OSCTXT * pctxt,
    const char ** string,
    OSBOOL rec,
    OSUINT32 startFlags,
    OSUINT32 endFlags )
```

This function will decode an ISO 8601 INTERVAL type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string variable to receive decoded value in string form (start/end).
<i>rec</i>	Decode recursive interval (Rn/).
<i>startFlags</i>	Set format flags of interval start: OSANY OSCENTURY OSYEAR OSMONTH OSWEEK OSDAY OSHOURS OSMINUTES OSSECONDS OSUTC OSDIFF n or OSDURATION. n - set digit number of fraction part.
<i>endFlags</i>	Set format flags of interval end.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.40 pd_isFragmented()

```
OSBOOL pd_isFragmented (
    OSCTXT * pctxt )
```

This function peeks at the open type length to determine if it is fragmented.

Parameters

<i>pctxt</i>	Pointer to a context structure.
--------------	---------------------------------

6.4.3.41 pd_Length()

```
int pd_Length (
    OSCTXT * pctxt,
    OSUINT32 * pvalue )
```

This function will decode a length determinant value.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to an unsigned integer variable to receive the decoded length value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.42 pd_Length64()

```
int pd_Length64 (
    OSCTXT * pctxt,
    OSSIZE * pvalue )
```

This function will decode a length determinant value. This variant support lengths up to 64 bits in size on 64-bit architectures.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to a size type variable to receive the decoded length value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.43 pd_ObjectIdentifier()

```
int pd_ObjectIdentifier (
    OSCTXT * pctxt,
    ASN1OBJID * pvalue )
```

This function decodes a value of the ASN.1 object identifier type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to receive decoded result. The ASN1OBJID structure contains an integer to hold the number of subidentifiers and an array to hold the subidentifier values.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.44 pd_OctetString()

```
int pd_OctetString (
    OSCTXT * pctxt,
    OSUINT32 * pnumocts,
    OSOCTET * buffer,
    OSUINT32 bufsiz )
```

This function will decode a value of the ASN.1 octet string type whose maximum size is known in advance. The ASN1C compiler generates a call to this function to decode octet string productions or elements that contain a size constraint.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pnumocts</i>	A pointer to an unsigned buffer of bufsiz octets to receive decoded data.
<i>buffer</i>	A pointer to a pre-allocated buffer of size octets to receive the decoded data.
<i>bufsiz</i>	The size of the buffer to receive the decoded result.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.45 pd_OctetString64()

```
int pd_OctetString64 (
    OSCTXT * pctxt,
    OSSIZE * pnumocts,
    OSOCTET * buffer,
    OSSIZE bufsiz )
```

This function will decode a value of the ASN.1 octet string type whose maximum size is known in advance. The ASN1C compiler generates a call to this function to decode octet string productions or elements that contain a size constraint. This variant of the function supports OCTET STRING's with sizes up to 64 bits in size on 64 bit systems.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pnumocts</i>	A pointer to an unsigned buffer of bufsiz octets to receive decoded data.
<i>buffer</i>	A pointer to a pre-allocated buffer of size octets to receive the decoded data.
<i>bufsiz</i>	The size of the buffer to receive the decoded result.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.46 pd_oid64()

```
int pd_oid64 (
    OSCTXT * pctxt,
    ASN1OID64 * pvalue )
```

This function decodes a value of the ASN.1 object identifier type using 64-bit subidentifiers.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to receive decoded result. The ASN1OID64 structure contains an integer to hold the number of subidentifiers and an array of 64-bit unsigned integers to hold the subidentifier values.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.47 pd_OpenType()

```
int pd_OpenType (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSSIZE * pnumocts )
```

This function will decode an ASN.1 open type. This used to be the ASN.1 ANY type, but now is used in a variety of applications requiring an encoding that can be interpreted by a decoder without prior knowledge of the type of the variable.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p2</i>	A pointer to an open type variable to receive the decoded data.
<i>pnumocts</i>	A pointer to an unsigned buffer of bufsiz octets to receive decoded data.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.48 pd_OpenTypeExt()

```
int pd_OpenTypeExt (
    OSCTXT * pctxt,
    const OSOCTET ** object_p2,
    OSSIZE * pnumocts )
```

This function will decode an ASN.1 open type extension. These are extra fields in a version-2 message that may be present after the ... extension marker. An open type structure (extElem1) is added to a message structure that contains an extension marker but no extension elements. The pd_OpenTypeExt function will populate this structure with the complete extension information (optional bit or choice index, length and data). A subsequent call to pe_OpenTypeExt will cause the saved extension fields to be included in a newly encoded message of the given type.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>object_p2</i>	A pointer to an open type variable to receive the decoded data.
<i>pnumocts</i>	A pointer to an unsigned buffer of bufsiz octets to receive decoded data.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.49 pd_Real()

```
int pd_Real (
    OSCTXT * pctxt,
    OSREAL * pvalue )
```

This function will decode a value of the binary encoded ASN.1 real type. This function provides support for the plus-infinity special real values.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all workings variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to a real variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

6.4.3.50 pd_Real10()

```
int pd_Real10 (
    OSCTXT * pctxt,
    const char ** ppvalue )
```

This function will decode a value of the decimal encoded ASN.1 real type.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all workings variables that must be maintained between function calls.
<i>ppvalue</i>	Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.51 pd_RelativeOID()

```
int pd_RelativeOID (
    OSCTXT * pctxt,
    ASN1OBJID * pvalue )
```

This function decodes a value of the ASN.1 RELATIVE-OID type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to receive decoded result. The ASN1OBJID structure contains an integer to hold the number of subidentifiers and an array to hold the subidentifier values.

Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

6.4.3.52 pd_SemiConsInt16()

```
int pd_SemiConsInt16 (
    OSCTXT * pctxt,
    OSINT16 * pvalue,
    OSINT64 lower )
```

This function will decode a semi-constrained 16-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as signed 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.53 pd_SemiConsInt64()

```
int pd_SemiConsInt64 (
    OSCTXT * pctxt,
    OSINT64 * pvalue,
    OSINT64 lower )
```

This function will decode a semi-constrained 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as signed 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.54 pd_SemiConsInt8()

```
int pd_SemiConsInt8 (  
    OSCTXT * pctxt,  
    OSINT8 * pvalue,  
    OSINT64 lower )
```

This function will decode a semi-constrained 8-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 8-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as signed 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.55 pd_SemiConsInteger()

```
int pd_SemiConsInteger (  
    OSCTXT * pctxt,  
    OSINT32 * pvalue,  
    OSINT64 lower )
```

This function will decode a semi-constrained integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as signed integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.56 pd_SemiConsUInt16()

```
int pd_SemiConsUInt16 (  
    OSCTXT * pctxt,  
    OSUINT16 * pvalue,  
    OSUINT64 lower )
```

This function will decode a semi-constrained unsigned 16-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.57 pd_SemiConsUInt16SignedBound()

```
int pd_SemiConsUInt16SignedBound (  
    OSCTXT * pctxt,  
    OSUINT16 * pvalue,  
    OSINT64 lower )
```

This function will decode a semi-constrained unsigned 16-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.58 pd_SemiConsUInt64()

```
int pd_SemiConsUInt64 (
    OSCTXT * pctxt,
    OSUINT64 * pvalue,
    OSUINT64 lower )
```

This function will decode a semi-constrained unsigned 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.59 pd_SemiConsUInt64SignedBound()

```
int pd_SemiConsUInt64SignedBound (
    OSCTXT * pctxt,
    OSUINT64 * pvalue,
    OSINT64 lower )
```

This function will decode a semi-constrained unsigned 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 64-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.60 pd_SemiConsUInt8()

```
int pd_SemiConsUInt8 (
    OSCTXT * pctxt,
    OSUINT8 * pvalue,
    OSUINT64 lower )
```

This function will decode a semi-constrained unsigned 8-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 8-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.61 pd_SemiConsUInt8SignedBound()

```
int pd_SemiConsUInt8SignedBound (
    OSCTXT * pctxt,
    OSUINT8 * pvalue,
    OSINT64 lower )
```

This function will decode a semi-constrained unsigned 8-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 8-bit integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.62 pd_SemiConsUnsigned()

```
int pd_SemiConsUnsigned (
    OSCTXT * pctxt,
    OSUINT32 * pvalue,
    OSUINT64 lower )
```

This function will decode a semi-constrained unsigned integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.63 pd_SemiConsUnsignedSignedBound()

```
int pd_SemiConsUnsignedSignedBound (
    OSCTXT * pctxt,
    OSINT32 * pvalue,
    OSINT64 lower )
```

This function will decode a semi-constrained unsigned integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned integer variable to receive decoded value.
<i>lower</i>	Lower range value, represented as unsigned integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.64 pd_SmallLength()

```
int pd_SmallLength (
    OSCTXT * pctxt,
    OSUINT32 * pvalue )
```

This function will decode a normally small length determinant as specified in 11.9 of the X.691 standard. This is a number that is expected to be small, but whose size is potentially unlimited due to the presence of an extension maker.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all workings variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to an unsigned integer value to receive decoded results.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.65 pd_SmallNonNegWholeNumber()

```
int pd_SmallNonNegWholeNumber (
    OSCTXT * pctxt,
    OSUINT32 * pvalue )
```

This function will decode a small non-negative whole number as specified in Section 10.6 of the X.691 standard. This is a number that is expected to be small, but whose size is potentially unlimited due to the presence of an extension maker.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all workings variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to an unsigned integer value to receive decoded results.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.66 pd_TimeStr()

```
int pd_TimeStr (
    OSCTXT * pctxt,
    const char ** string,
    OSUINT32 flags )
```

This function will decode ISO 8601 TIME types.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string variable to receive decoded value in string form (HH:MM:SS) and ect.
<i>flags</i>	Set of flags: OSHOURS OSMINUTES OSSECONDS OSUTC OSDIFF n. n - set digit number of fraction part.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.67 pd_UnconsInt16()

```
int pd_UnconsInt16 (
    OSCTXT * pctxt,
    OSINT16 * pvalue )
```

This function will decode an unconstrained 16-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 16-bit integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.68 pd_UnconsInt64()

```
int pd_UnconsInt64 (
    OSCTXT * pctxt,
    OSINT64 * pvalue )
```

This function will decode an unconstrained 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 64-bit integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.69 pd_UnconsInt8()

```
int pd_UnconsInt8 (
    OSCTXT * pctxt,
    OSINT8 * pvalue )
```

This function will decode an unconstrained 8-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to 8-bit integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.70 pd_UnconsInteger()

```
int pd_UnconsInteger (
    OSCTXT * pctxt,
    OSINT32 * pvalue )
```

This function will decode an unconstrained integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.71 pd_UnconsLength()

```
int pd_UnconsLength (
    OSCTXT * pctxt,
    OSUINT32 * pvalue )
```

This function will decode an unconstrained length value or a length value with upper bound > 64k.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.72 pd_UnconsLength64()

```
EXTPERMETHOD int pd_UnconsLength64 (  
    OSCTXT * pctxt,  
    OSSIZE * pvalue )
```

This function will decode an unconstrained length value or a length value with upper bound > 64k. This variant supports lengths up to 64 bits in size.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to size type variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.73 pd_UnconsUInt16()

```
int pd_UnconsUInt16 (  
    OSCTXT * pctxt,  
    OSUINT16 * pvalue )
```

This function will decode an unconstrained integer into an unsigned 16-bit integer. An error is returned if the value being decoded cannot be represented by *pvalue (it is negative or too large).

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 16-bit integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.74 pd_UnconsUInt64()

```
int pd_UnconsUInt64 (
    OSCTXT * pctxt,
    OSUINT64 * pvalue )
```

This function will decode an unconstrained integer into an unsigned 64-bit integer. An error is returned if the value being decoded cannot be represented by **pvalue* (it is negative or too large).

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 64-bit integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.75 pd_UnconsUInt8()

```
int pd_UnconsUInt8 (
    OSCTXT * pctxt,
    OSUINT8 * pvalue )
```

This function will decode an unconstrained integer into an unsigned 8-bit integer. An error is returned if the value being decoded cannot be represented by **pvalue* (it is negative or too large).

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned 8-bit integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.76 pd_UnconsUnsigned()

```
int pd_UnconsUnsigned (
    OSCTXT * pctxt,
    OSUINT32 * pvalue )
```

This function will decode an unconstrained integer into an unsigned type. An error is returned if the value to be decoded cannot be represented by *pvalue (it is either negative or too large).

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to unsigned integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.77 pd_UniversalString()

```
int pd_UniversalString (
    OSCTXT * pctxt,
    ASN1UniversalString * pvalue,
    Asn132BitCharSet * permCharSet )
```

This function will decode a variable of the ASN.1 32-bit character string. This differs from the decode routines for the character strings previously described because the universal string type is based on 32-bit characters. A 32-bit character string is modeled using an array of unsigned integers.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	Pointer to character string structure to receive the decoded result. The structure includes a count field containing the number of characters and an array of unsigned short integers to hold the 32-bit character values.
<i>permCharSet</i>	A pointer to the constraining character set. This contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,

- negative return value is error.

6.4.3.78 pd_VarWidthCharString()

```
int pd_VarWidthCharString (
    OSCTXT * pctxt,
    const char ** pvalue )
```

This function will decode a variable of the ASN.1 character string.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to a character pointer variable to receive the decoded result. Dynamic memory is allocated for the variable using the ::rtxMemAlloc function.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.79 pd_YearInt()

```
int pd_YearInt (
    OSCTXT * pctxt,
    OSINT32 * pvalue )
```

This function will decode an ISO 8601 YEAR type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to integer variable to receive decoded value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.80 uperDecConstrFixedLenString()

```
int uperDecConstrFixedLenString (
    OSCTXT * pctxt,
    char * strbuf,
    size_t bufsiz,
    const char * charSet,
    OSUINT32 nbits,
    OSUINT32 canSetBits )
```

This function decodes a constrained string value into a fixed-size buffer. This version supports unaligned PER only. It allows all of the required permitted alphabet constraint parameters to be passed in as arguments.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>strbuf</i>	Pointer to character array to receive decoded string.
<i>bufsiz</i>	Size of <i>strbuf</i> character array.
<i>charSet</i>	String containing permitted alphabet character set. Can be null if no character set was specified.
<i>nbits</i>	Number of bits in a character set character (unaligned).
<i>canSetBits</i>	Number of bits in a character from the canonical set representing this string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.4.3.81 uperDecConstrString()

```
int uperDecConstrString (
    OSCTXT * pctxt,
    const char ** string,
    const char * charSet,
    OSUINT32 nbits,
    OSUINT32 canSetBits )
```

This function decodes a constrained string value. This version supports unaligned PER only. It allows all of the required permitted alphabet constraint parameters to be passed in as arguments.

Parameters

<i>pctxt</i>	Pointer to context block structure.
--------------	-------------------------------------

Parameters

<i>string</i>	Pointer to const char* to receive decoded string. Memory will be allocated for this variable using internal memory management functions.
<i>charSet</i>	String containing permitted alphabet character set. Can be null if no character set was specified.
<i>nbits</i>	Number of bits in a character set character (unaligned).
<i>canSetBits</i>	Number of bits in a character from the canonical set representing this string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5 PER C Encode Functions.

Macros

- #define `pe_bit`(pctx, value) `rtxEncBit`(pctx,value)
- #define `pe_bits`(pctx, value, nbits) `rtxEncBits`(pctx,value,nbits)
- #define `pe_CheckBuffer`(pctx, nbytes) `rtxCheckOutputBuffer`(pctx,nbytes)
- #define `pe_ConsInteger`(pctx, value, lower, upper) `pe_ConsInt64`(pctx, value, lower, upper)
- #define `pe_ConsUnsigned`(pctx, value, lower, upper) `pe_ConsUInt64`(pctx, value, lower, upper)
- #define `pe_ConsUnsignedSignedBound`(pctx, value, lower, upper) `pe_ConsUInt64SignedBound`(pctx, value, lower, upper)
- #define `pe_octets`(pctx, pvalue, nbits) `rtxEncBitsFromArray`(pctx,pvalue,nbits)
- #define `pe_SemiConsInteger`(pctx, value, lower) `pe_SemiConsInt64`(pctx, value, lower)
- #define `pe_SemiConsUnsigned`(pctx, value, lower) `pe_SemiConsUInt64`(pctx, value, lower)

Functions

- int `pe_16BitConstrainedString` (OSCTXT *pctx, Asn116BitCharString value, Asn116BitCharSet *pCharSet)
- int `pe_32BitConstrainedString` (OSCTXT *pctx, Asn132BitCharString value, Asn132BitCharSet *pCharSet)
- int `pe_2sCompBinInt` (OSCTXT *pctx, OSINT32 value)
- int `pe_2sCompBinInt64` (OSCTXT *pctx, OSINT64 value)
- int `pe_aligned_octets` (OSCTXT *pctx, OSOCTET *pvalue, OSUINT32 nocts)
- int `pe_BigInteger` (OSCTXT *pctx, const char *pvalue)
- int `pe_bits64` (OSCTXT *pctx, OSUINT64 value, OSUINT32 nbits)
- int `pe_BitString` (OSCTXT *pctx, OSSIZE numbits, const OSOCTET *data)
- int `pe_BitString32` (OSCTXT *pctx, ASN1BitStr32 *pbitstr, OSUINT32 lower, OSUINT32 upper)
- EXTPERMETHOD int `pe_BitStringExt` (OSCTXT *pctx, OSSIZE numbits, const OSOCTET *data, OSSIZE dataSize, const OSOCTET *extData)
- int `pe_BMPString` (OSCTXT *pctx, ASN1BMPString value, Asn116BitCharSet *permCharSet)
- int `pe_UniversalString` (OSCTXT *pctx, ASN1UniversalString value, Asn132BitCharSet *permCharSet)
- int `pe_byte_align` (OSCTXT *pctx)
- int `pe_ChoiceTypeExt` (OSCTXT *pctx, OSUINT32 numocts, const OSOCTET *data)
- int `pe_ConsInt64` (OSCTXT *pctx, OSINT64 value, OSINT64 lower, OSINT64 upper)
- int `pe_ConstrainedString` (OSCTXT *pctx, const char *string, Asn1CharSet *pCharSet)
- int `pe_ConstrainedStringEx` (OSCTXT *pctx, const char *string, const char *charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int `pe_ConsUInt64` (OSCTXT *pctx, OSUINT64 value, OSUINT64 lower, OSUINT64 upper)
- int `pe_ConsUInt64SignedBound` (OSCTXT *pctx, OSUINT64 value, OSINT64 lower, OSINT64 upper)
- int `pe_ConsWholeNumber` (OSCTXT *pctx, OSUINT32 adjusted_value, OSUINT32 range_value)
- int `pe_ConsWholeNumber64` (OSCTXT *pctx, OSUINT64 adjusted_value, OSUINT64 range_value)
- int `pe_DateStr` (OSCTXT *pctx, const char *string, OSUINT32 flags)
- int `pe_DateTimeStr` (OSCTXT *pctx, const char *string, OSUINT32 flags)
- int `pe_Duration` (OSCTXT *pctx, const char *string, OSBOOL rec)
- OSUINT32 `pe_GetIntLen` (OSUINT32 value)
- size_t `pe_GetMsgBitCnt` (OSCTXT *pctx)
- OSOCTET * `pe_GetMsgPtr` (OSCTXT *pctx, OSINT32 *pLength)
- OSOCTET * `pe_GetMsgPtrU` (OSCTXT *pctx, OSUINT32 *pLength)
- OSOCTET * `pe_GetMsgPtr64` (OSCTXT *pctx, OSSIZE *pLength)
- int `pe_Interval` (OSCTXT *pctx, const char *string, OSBOOL rec, OSUINT32 startFlags, OSUINT32 endFlags)

- int `pe_Length` (OSCTXT *pctx, OSSIZE value)
- int `pe_NonNegBinInt` (OSCTXT *pctx, OSUINT32 value)
- int `pe_NonNegBinInt64` (OSCTXT *pctx, OSUINT64 value)
- int `pe_ObjectIdentifier` (OSCTXT *pctx, ASN1OBJID *pvalue)
- int `pe_oid64` (OSCTXT *pctx, ASN1OID64 *pvalue)
- int `pe_RelativeOID` (OSCTXT *pctx, ASN1OBJID *pvalue)
- int `pe_OcetCodeString` (OSCTXT *pctx, OSSIZE numocts, const OSOCTET *data)
- int `pe_OpenType` (OSCTXT *pctx, OSSIZE numocts, const OSOCTET *data)
- int `pe_OpenTypeEnd` (OSCTXT *pctx, OSUINT32 pos, void *pPerField)
- int `pe_OpenTypeExt` (OSCTXT *pctx, OSRTDList *pElemList)
- int `pe_OpenTypeExtBits` (OSCTXT *pctx, OSRTDList *pElemList)
- int `pe_OpenTypeStart` (OSCTXT *pctx, OSUINT32 *pPos, void **ppPerField)
- int `pe_Real` (OSCTXT *pctx, OSREAL value)
- int `pe_SmallNonNegWholeNumber` (OSCTXT *pctx, OSUINT32 value)
- int `pe_SmallLength` (OSCTXT *pctx, OSSIZE value)
- int `pe_SemiConsInt64` (OSCTXT *pctx, OSINT64 value, OSINT64 lower)
- int `pe_SemiConsUnsignedSignedBound` (OSCTXT *pctx, OSUINT32 value, OSINT64 lower)
- int `pe_SemiConsUInt64` (OSCTXT *pctx, OSUINT64 value, OSUINT64 lower)
- int `pe_SemiConsUInt64SignedBound` (OSCTXT *pctx, OSUINT64 value, OSINT64 lower)
- int `pe_TimeStr` (OSCTXT *pctx, const char *string, OSUINT32 flags)
- int `pe_UnconsLength` (OSCTXT *pctx, OSSIZE value)
- int `pe_UnconsInteger` (OSCTXT *pctx, OSINT32 value)
- int `pe_UnconsInt64` (OSCTXT *pctx, OSINT64 value)
- int `pe_UnconsUnsigned` (OSCTXT *pctx, OSUINT32 value)
- int `pe_UnconsUInt64` (OSCTXT *pctx, OSUINT64 value)
- int `pe_VarWidthCharString` (OSCTXT *pctx, const char *value)
- int `pe_YearInt` (OSCTXT *pctx, OSINT32 value)
- int `pe_Real10` (OSCTXT *pctx, const char *pvalue)
- int `uperEncConstrString` (OSCTXT *pctx, const char *string, const char *charSet, OSUINT32 nbits, OSUINT32 canSetBits)

6.5.1 Detailed Description

The Per low-level encode functions handle the PER encoding of the primitive ASN.1 data types. Calls to these functions are assembled in the C source code generated by the ASN1C compiler to accomplish the encoding of complex ASN.1 structures. These functions are also directly callable from within a user's application program if the need to accomplish a low level encoding function exists.

The procedure to call a low-level encode function is the same as the procedure to call a compiler generated encode function described above. The `pu_newContext` function must first be called to set a pointer to the buffer into which the variable is to be encoded. A static encode buffer is specified by assigning a pointer to a buffer and a buffer size. Setting the buffer address to NULL and buffer size to 0 specifies a dynamic buffer. The encode function is then invoked. The result of the encoding will start at the beginning of the specified buffer, or, if a dynamic buffer was used, only be obtained by calling `pe_GetMsgPtr`. The length of the encoded compound is obtained by calling `pe_GetMsgLen`.

6.5.2 Macro Definition Documentation

6.5.2.1 pe_bit

```
#define pe_bit(  
    pctxt,  
    value ) rtxEncBit(pctxt,value)
```

This function will encode a variable of the ASN.1 BOOLEAN type in single bit,

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	The BOOLEAN value to be encoded.

6.5.2.2 pe_bits

```
#define pe_bits(  
    pctxt,  
    value,  
    nbits ) rtxEncBits(pctxt,value,nbits)
```

This function encodes multiple bits.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Unsigned integer containing the bits to be encoded.
<i>nbits</i>	Number of bits in value to encode.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.3 pe_CheckBuffer

```
#define pe_CheckBuffer(  
    pctxt,  
    nbytes ) rtxCheckOutputBuffer(pctxt,nbytes)
```

This function will determine if the given number of bytes will fit in the encode buffer. If not, either the buffer is expanded (if it is a dynamic buffer) or an error is signaled.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>nbytes</i>	Number of bytes of space required to hold the variable to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.4 pe_ConsInteger

```
#define pe_ConsInteger(  
    pctxt,  
    value,  
    lower,  
    upper ) pe_ConsInt64(pctxt, value, lower, upper)
```

This function encodes an integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.5 pe_ConsUnsigned

```
#define pe_ConsUnsigned(  
    pctxt,  
    value,
```

```

    lower,
    upper ) pe_ConsumInt64(pctxt, value, lower, upper)

```

This function encodes an unsigned integer constrained either by a value or value range constraint. The constrained unsigned integer option is used if:

1. The lower value of the range is ≥ 0 , and 2. The upper value of the range is $\geq \text{MAXINT}$

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.6 `pe_ConsumSignedBound`

```

#define pe_ConsumSignedBound(
    pctxt,
    value,
    lower,
    upper ) pe_ConsumInt64SignedBound(pctxt, value, lower, upper)

```

This function encodes an unsigned integer constrained either by a value or value range constraint. The constrained unsigned integer option is used if:

1. The lower value of the range is ≥ 0 , and 2. The upper value of the range is $\geq \text{MAXINT}$

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>lower</i>	Lower range value.
<i>upper</i>	Upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.7 pe_octets

```
#define pe_octets(  
    pctxt,  
    pvalue,  
    nbits ) rtxEncBitsFromArray(pctxt,pvalue,nbits)
```

This function will encode an array of octets. The Octets will be encoded unaligned starting at the current bit offset within the encode buffer.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	A pointer to an array of octets to encode
<i>nbits</i>	The number of Octets to encode

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.8 pe_SemiConsInteger

```
#define pe_SemiConsInteger(  
    pctxt,  
    value,  
    lower ) pe_SemiConsInt64(pctxt, value, lower)
```

This function encodes a semi-constrained integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>lower</i>	Lower range value, represented as signed integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.2.9 pe_SemiConsUnsigned

```
#define pe_SemiConsUnsigned(  
    pctxt,  
    value,  
    lower ) pe_SemiConsUInt64(pctxt, value, lower)
```

This function encodes an semi-constrained unsigned integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>lower</i>	Lower range value, represented as unsigned integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3 Function Documentation

6.5.3.1 pe_16BitConstrainedString()

```
int pe_16BitConstrainedString (  
    OSCTXT * pctxt,  
    Asn116BitCharString value,  
    Asn116BitCharSet * pCharSet )
```

This function will encode a constrained ASN.1 character string. This function is normally not called directly but rather is called from Useful Type Character String encode functions that deal with 16-bit strings. The only function that does that in this release is the `pe_BMPString` function.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Character string to be encoded. The structure includes a count field containing the number of characters to encode and an array of unsigned short integers to hold the 16-bit characters to be encoded.
<i>pCharSet</i>	Pointer to the constraining character set. The contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.2 pe_2sCompBinInt()

```
int pe_2sCompBinInt (
    OSCTXT * pctxt,
    OSINT32 value )
```

This function encodes a two's complement binary integer as specified in Section 10.4 of the X.691 standard.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Signed integer value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.3 pe_2sCompBinInt64()

```
int pe_2sCompBinInt64 (
    OSCTXT * pctxt,
    OSINT64 value )
```

This function encodes a two's complement binary 64-bit integer as specified in Section 10.4 of the X.691 standard.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Signed 64-bit integer value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.4 pe_32BitConstrainedString()

```
int pe_32BitConstrainedString (
    OSCTXT * pctxt,
    Asn132BitCharString value,
    Asn132BitCharSet * pCharSet )
```

This function will encode a constrained ASN.1 character string. This function is normally not called directly but rather is called from Useful Type Character String encode functions that deal with 32-bit strings. The only function that does that in this release is the pe_UniversalString function.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Character string to be encoded. The structure includes a count field containing the number of characters to encode and an array of unsigned short integers to hold the 32-bit characters to be encoded.
<i>pCharSet</i>	Pointer to the constraining character set. The contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.5 pe_aligned_octets()

```
int pe_aligned_octets (
    OSCTXT * pctxt,
```

```
OSOCKET * pvalue,  
OSUINT32 nocts )
```

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	A pointer to a character string containing the value to be encoded.
<i>nocts</i>	The number of octets.

6.5.3.6 pe_BigInteger()

```
int pe_BigInteger (
    OSCTXT * pctxt,
    const char * pvalue )
```

The `pe_BigInteger` function will encode a variable of the ASN.1 INTEGER type. In this case, the integer is assumed to be of a larger size than can fit in a C or C++ long type (normally 32 or 64 bits). For example, parameters used to calculate security values are typically larger than these sizes. Items of this type are stored in character string constant variables. They can be represented as decimal strings (with no prefixes), as hexadecimal strings starting with a "0x" prefix, as octal strings starting with a "0o" prefix or as binary strings starting with a "0b" prefix. Other radices currently are not supported. It is highly recommended to use the hexadecimal or binary strings for better performance.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	A pointer to a character string containing the value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.7 pe_bits64()

```
int pe_bits64 (
    OSCTXT * pctxt,
    OSUINT64 value,
    OSUINT32 nbits )
```

This function encodes multiple bits, using unsigned 64-bit integer to hold bits.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Unsigned 64-bit integer containing the bits to be encoded.
<i>nbits</i>	Number of bits in value to encode. 77

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.8 pe_BitString()

```
int pe_BitString (
    OSCTXT * pctxt,
    OSSIZE numbits,
    const OSOCTET * data )
```

This function will encode a value of the ASN.1 bit string type.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>numbits</i>	The number of bits in the string to be encoded.
<i>data</i>	Pointer to the bit string data to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.9 pe_BitStringExt()

```
EXTPERMETHOD int pe_BitStringExt (
    OSCTXT * pctxt,
    OSSIZE numbits,
    const OSOCTET * data,
    OSSIZE dataSize,
    const OSOCTET * extData )
```

This function will encode a value of the ASN.1 bit string type. In this case, the bit string has an extended size and may contain extension data.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>numbits</i>	The total number of bits in the string to be encoded (root + extension).
<i>data</i>	Pointer to the root bit string data to be encoded.
<i>dataSize</i>	Size, in octets, of the root bit string data.
<i>extData</i>	Pointer to extension bit string data to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.10 pe_BMPString()

```
int pe_BMPString (
    OSCTXT * pctxt,
    ASN1BMPString value,
    Asn116BitCharSet * permCharSet )
```

This function will encode a variable of the ASN.1 BMP character string. This differs from the encode routines for the character strings previously described in that the BMP string type is based on 16-bit characters. A 16-bit character string is modeled using an array of unsigned short integers.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Character string to be encoded. This structure includes a count field containing the number of characters to encode and an array of unsigned short integers to hold the 16-bit characters to be encoded.
<i>permCharSet</i>	Pointer to the constraining character set. This contains an array containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.11 pe_byte_align()

```
int pe_byte_align (
    OSCTXT * pctxt )
```

This function will position the encode bit cursor on the next byte boundary.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

Referenced by ASN1PEREncodeBuffer::byteAlign().

6.5.3.12 pe_ChoiceTypeExt()

```
int pe_ChoiceTypeExt (
    OSCTXT * pctxt,
    OSUINT32 numocts,
    const OSOCTET * data )
```

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>numocts</i>	Number of octets in the string to be encoded.
<i>data</i>	Pointer to octet string data to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.13 pe_ConsInt64()

```
int pe_ConsInt64 (
    OSCTXT * pctxt,
    OSINT64 value,
    OSINT64 lower,
    OSINT64 upper )
```

This function encodes a 64-bit integer constrained either by a value or value range constraint.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as 64-bit integer.
<i>lower</i>	Lower range value, represented as 64-bit integer.
<i>upper</i>	Upper range value, represented as 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.14 pe_ConstrainedString()

```
int pe_ConstrainedString (
    OSCTXT * pctxt,
    const char * string,
    Asn1CharSet * pCharSet )
```

This function encodes a constrained string value. This is a deprecated version of the function provided for backward compatibility.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string to be encoded.
<i>pCharSet</i>	Pointer to a character set descriptor structure.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.15 pe_ConstrainedStringEx()

```
int pe_ConstrainedStringEx (
    OSCTXT * pctxt,
    const char * string,
    const char * charSet,
    OSUINT32 abits,
    OSUINT32 ubits,
    OSUINT32 canSetBits )
```

This function encodes a constrained string value. This version of the function allows all of the required permitted alphabet constraint parameters to be passed in as arguments.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string to be encoded.
<i>charSet</i>	String containing permitted alphabet character set. Can be null if no character set was specified.
<i>abits</i>	Number of bits in a character set character (aligned).
<i>ubits</i>	Number of bits in a character set character (unaligned).
<i>canSetBits</i>	Number of bits in a character from the canonical set representing this string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.16 pe_ConsUInt64()

```
int pe_ConsUInt64 (
    OSCTXT * pctxt,
    OSUINT64 value,
    OSUINT64 lower,
    OSUINT64 upper )
```

This function encodes an unsigned 64-bit integer constrained either by a value or value range constraint. The constrained unsigned integer option is used if:

1. The lower value of the range is ≥ 0 , and 2. The upper value of the range is $\geq \text{MAXINT}$

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as unsigned 64-bit integer.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.
<i>upper</i>	Upper range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.17 `pe_ConsUInt64SignedBound()`

```
int pe_ConsUInt64SignedBound (
    OSCTXT * pctxt,
    OSUINT64 value,
    OSINT64 lower,
    OSINT64 upper )
```

This function encodes an unsigned 64-bit integer constrained either by a value or value range constraint. The constrained unsigned integer option is used if:

1. The lower value of the range is ≥ 0 , and 2. The upper value of the range is $\geq \text{MAXINT}$

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as unsigned 64-bit integer.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.
<i>upper</i>	Upper range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.18 pe_ConsWholeNumber()

```
int pe_ConsWholeNumber (
    OSCTXT * pctxt,
    OSUINT32 adjusted_value,
    OSUINT32 range_value )
```

This function encodes a constrained whole number as specified in Section 10.5 of the X.691 standard.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>adjusted_value</i>	Unsigned adjusted integer value to be encoded. The adjustment is done by subtracting the lower value of the range from the value to be encoded.
<i>range_value</i>	Unsigned integer value specifying the total size of the range. This is obtained by subtracting the lower range value from the upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.19 pe_ConsWholeNumber64()

```
int pe_ConsWholeNumber64 (
    OSCTXT * pctxt,
    OSUINT64 adjusted_value,
    OSUINT64 range_value )
```

This function encodes a constrained whole number as specified in Section 10.5 of the X.691 standard, represented as 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>adjusted_value</i>	Unsigned adjusted integer value to be encoded. The adjustment is done by subtracting the lower value of the range from the value to be encoded.
<i>range_value</i>	Unsigned integer value specifying the total size of the range. This is obtained by subtracting the lower range value from the upper range value.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.20 pe_DateStr()

```
int pe_DateStr (
    OSCTXT * pctxt,
    const char * string,
    OSUINT32 flags )
```

This function will encode an ISO 8601 DATE types.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Character string variable containing value to be encoded in string form (YYYY:MM:DD) and ect.
<i>flags</i>	Set of flags: OSANY OSCENTURY OSYEAR OSMONTH OSWEEK OSDAY.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.21 pe_DateTimeStr()

```
int pe_DateTimeStr (
    OSCTXT * pctxt,
    const char * string,
    OSUINT32 flags )
```

This function will encode an ISO 8601 DATE-TIME types.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Character string variable containing value to be encoded in string form (YYYY-MM-DDTHH:MM:SS) and ect.
<i>flags</i>	Set of flags: OSANY OSCENTURY OSYEAR OSMONTH OSWEEK OSDAY OSHOURS OSMINUTES OSSECONDS OSUTC OSDIFF n. n - set digit number of fraction part.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.22 pe_Duration()

```
int pe_Duration (
    OSCTXT * pctxt,
    const char * string,
    OSBOOL rec )
```

This function will encode an ISO 8601 DURATION types.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Character string variable containing value to be encoded in string form (PnYnMnDTnHnMnS).
<i>rec</i>	Encode recursive interval (Rn/).

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.23 pe_GetIntLen()

```
OSUINT32 pe_GetIntLen (
    OSUINT32 value )
```

Parameters

<i>value</i>	Length value to be encoded.
--------------	-----------------------------

6.5.3.24 pe_GetMsgBitCnt()

```
size_t pe_GetMsgBitCnt (
```

```
OSCTXT * pctx )
```

Parameters

<i>pctx</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
-------------	--

Referenced by ASN1PEREncodeBuffer::getMsgBitCnt().

6.5.3.25 pe_GetMsgPtr()

```
OSOCKET* pe_GetMsgPtr (
    OSCTXT * pctx,
    OSINT32 * pLength )
```

This function will return the message pointer and length of an encoded message. This function is called after a compiler generated encode function to get the pointer and length of the message. It is normally used when dynamic encoding is specified because the message pointer is not known until encoding is complete. If static encoding is used, the message starts at the beginning of the specified buffer and the pe_GetMsgLen function can be used to obtain the length of the message.

Parameters

<i>pctx</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pLength</i>	Pointer to variable to receive length of the encoded message.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.26 pe_GetMsgPtr64()

```
OSOCKET* pe_GetMsgPtr64 (
    OSCTXT * pctx,
    OSIZE * pLength )
```

This version of pe_GetMsgPtr return the length as a size-typed value rather than a signed integer.

See also

[pe_GetMsgPtr](#)

6.5.3.27 pe_GetMsgPtrU()

```
OSOCKET* pe_GetMsgPtrU (
    OSCTXT * pctxt,
    OSUINT32 * pLength )
```

This version of `pe_GetMsgPtr` return the length as a 32-bit unsigned rather than a signed integer.

See also

[pe_GetMsgPtr](#)

6.5.3.28 pe_Interval()

```
int pe_Interval (
    OSCTXT * pctxt,
    const char * string,
    OSBOOL rec,
    OSUINT32 startFlags,
    OSUINT32 endFlags )
```

This function will encode an ISO 8601 INTERVAL type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Character string variable containing value to be encoded in string form (start/end).
<i>rec</i>	Encode recursive interval (Rn/).
<i>startFlags</i>	Set format flags of interval start: OSANY OSCENTURY OSYEAR OSMONTH OSWEEK OSDAY OSHOURS OSMINUTES OSSECONDS OSUTC OSDIFF n or OSDURATION. n - set digit number of fraction part.
<i>endFlags</i>	Set format flags of interval end.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.29 pe_Length()

```
int pe_Length (
    OSCTXT * pctxt,
    OSSIZE value )
```

This function will encode a length determinant value.

Parameters

<i>pctxt</i>	Pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Length value to be encoded.

Returns

If successful, then one of the following values:

- if *value* \geq 16384, then fragmentation is required. The return will be the number of values to encode in the current fragment. The return will be a multiple of 16384, with the following constraints: $16384 \leq$ return \leq 65536 return \leq *value*
- *value* (the given length) Otherwise, a negative error status code.

6.5.3.30 pe_NonNegBinInt()

```
int pe_NonNegBinInt (
    OSCTXT * pctxt,
    OSUINT32 value )
```

This function encodes a non-negative binary integer as specified in Section 10.3 of the X.691 standard.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Unsigned integer value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.31 pe_NonNegBinInt64()

```
int pe_NonNegBinInt64 (
    OSCTXT * pctxt,
    OSUINT64 value )
```

This function encodes a non-negative binary 64-bit integer as specified in Section 10.3 of the X.691 standard.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Unsigned 64-bit integer value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.32 pe_ObjectIdentifier()

```
int pe_ObjectIdentifier (
    OSCTXT * pctxt,
    ASN1OBJID * pvalue )
```

This function encodes a value of the ASN.1 object identifier type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded. The ASN1OBJID structure contains a numids fields to hold the number of subidentifiers and an array to hold the subidentifier values.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.33 pe_OctetString()

```
int pe_OctetString (
    OSCTXT * pctxt,
    OSSIZE numocts,
    const OSOCTET * data )
```

This function will encode a value of the ASN.1 octet string type.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>numocts</i>	Number of octets in the string to be encoded.
<i>data</i>	Pointer to octet string data to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.34 pe_oid64()

```
int pe_oid64 (
    OSCTXT * pctxt,
    ASN1OID64 * pvalue )
```

This function encodes a value of the ASN.1 object identifier type, using 64-bit subidentifiers.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded. The ASN1OID64 structure contains a numids fields to hold the number of subidentifiers and an array of unsigned 64-bit integers to hold the subidentifier values.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.35 pe_OpenType()

```
int pe_OpenType (
    OSCTXT * pctxt,
    OSSIZE numocts,
    const OSOCTET * data )
```

This function will encode an ASN.1 open type. This used to be the ANY type, but now is used in a variety of applications requiring an encoding that can be interpreted by a decoder without a prior knowledge of the type of the variable.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>numocts</i>	Number of octets in the string to be encoded.
<i>data</i>	Pointer to octet string data to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.36 pe_OpenTypeEnd()

```
int pe_OpenTypeEnd (
    OSCTXT * pctxt,
    OSUINT32 pos,
    void * pPerField )
```

This function will finish encoding extension in ASN.1 open type. It will write open type length to saved position. If required function do fragmentation of open type data.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pos</i>	Position that was saved in pe_OpenTypeStart function.
<i>pPerField</i>	A pointer to bit field that was saved in pe_OpenTypeStart function.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.37 pe_OpenTypeExt()

```
int pe_OpenTypeExt (
    OSCTXT * pctxt,
    OSRTDList * pElemList )
```

This function will encode an ASN.1 open type extension. An open type extension field is the data that potentially resides after the ... marker in a version-1 message. The open type structure contains a complete encoded bit set including option element bits or choice index, length, and data. Typically, this data is populated when a version-1 system decodes a version-2 message. The extension fields are retained and can then be re-encoded if a new message is to be sent out (for example, in a store and forward system).

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pElemList</i>	A pointer to the open type to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.38 pe_OpenTypeExtBits()

```
int pe_OpenTypeExtBits (
    OSCTXT * pctxt,
    OSRTDList * pElemList )
```

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pElemList</i>	A pointer to the open type to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.39 pe_OpenTypeStart()

```
int pe_OpenTypeStart (
    OSCTXT * pctxt,
```

```
OSUINT32 * pPos,  
void ** ppPerField )
```

This function will start encoding extension in ASN.1 open type. It will reserve place to open type length and return current buffer position.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pPos</i>	A pointer to return current buffer position.
<i>ppPerField</i>	A pointer to to return current bit field record.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.40 pe_Real()

```
int pe_Real (
    OSCTXT * pctxt,
    OSREAL value )
```

This function will encode a value of the ASN.1 real type. This function provides support for the plus-infinity and minus-infinity special real values. Use the `rtxGetPlusInfinity` or `rtxGetMinusInfinity` functions to get these special values.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Value to be encoded. Special real values plus and minus infinity are encoded by using the <code>rtxGetPlusInfinity</code> and the <code>rtxGetMinusInfinity</code> functions to se the real value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.41 pe_Real10()

```
int pe_Real10 (
    OSCTXT * pctxt,
    const char * pvalue )
```

This function will encode a number from character string to ASN.1 real type using decimal encoding. Number may be represented in integer, decimal, and exponent formats.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>pvalue</i>	Value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.42 pe_RelativeOID()

```
int pe_RelativeOID (
    OSCTXT * pctxt,
    ASN1OBJID * pvalue )
```

This function encodes a value of the ASN.1 RELATIVE-OID type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>pvalue</i>	Pointer to value to be encoded. The ASN1OBJID structure contains a numids fields to hold the number of subidentifiers and an array to hold the subidentifier values.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.43 pe_SemiConsInt64()

```
int pe_SemiConsInt64 (
    OSCTXT * pctxt,
    OSINT64 value,
    OSINT64 lower )
```

This function encodes an semi-constrained 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as 64-bit integer.
<i>lower</i>	Lower range value, represented as signed 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.44 `pe_SemiConsUInt64()`

```
int pe_SemiConsUInt64 (
    OSCTXT * pctxt,
    OSUINT64 value,
    OSUINT64 lower )
```

This function encodes an semi-constrained unsigned 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as unsigned 64-bit integer.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.45 `pe_SemiConsUInt64SignedBound()`

```
int pe_SemiConsUInt64SignedBound (
    OSCTXT * pctxt,
    OSUINT64 value,
    OSINT64 lower )
```

This function encodes an semi-constrained unsigned 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as unsigned 64-bit integer.
<i>lower</i>	Lower range value, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.46 `pe_SemiConsUnsignedSignedBound()`

```
int pe_SemiConsUnsignedSignedBound (
    OSCTXT * pctxt,
    OSUINT32 value,
    OSINT64 lower )
```

This function encodes an semi-constrained unsigned integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.
<i>lower</i>	Lower range value, represented as unsigned integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.47 `pe_SmallLength()`

```
int pe_SmallLength (
    OSCTXT * pctxt,
    OSSIZE value )
```

This function will encode a normally small length as specified in Section 11.9 of the X.691 standard. This is a number that is expected to be small, but whose size is potentially unlimited due to the presence of an extension marker.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An unsigned integer value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.48 `pe_SmallNonNegWholeNumber()`

```
int pe_SmallNonNegWholeNumber (
    OSCTXT * pctxt,
    OSUINT32 value )
```

This function will encode a small, non-negative whole number as specified in Section 10.6 of the X.691 standard. This is a number that is expected to be small, but whose size is potentially unlimited due to the presence of an extension marker.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	An unsigned integer value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.49 `pe_TimeStr()`

```
int pe_TimeStr (
    OSCTXT * pctxt,
    const char * string,
    OSUINT32 flags )
```

This function will encode an ISO 8601 TIME types.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Character string variable containing value to be encoded in string form (HH:MM:SS) and ect.
<i>flags</i>	Set of flags: OSHOURS OSMINUTES OSSECONDS OSUTC OSDIFF n. n - set digit number of fraction part.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.50 pe_UnconsInt64()

```
int pe_UnconsInt64 (
    OSCTXT * pctxt,
    OSINT64 value )
```

This function encodes an unconstrained 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.51 pe_UnconsInteger()

```
int pe_UnconsInteger (
    OSCTXT * pctxt,
    OSINT32 value )
```

This function encodes an unconstrained integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.52 `pe_UnconsLength()`

```
int pe_UnconsLength (
    OSCTXT * pctxt,
    OSSIZE value )
```

This function encodes an unconstrained length value.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Value to be encoded.

Returns

If successful, the encoded number of bytes is returned; otherwise, a negative error status code. Note that in the non-error case, this will always be equal to value unless a fragmented length is encoded.

6.5.3.53 `pe_UnconsUInt64()`

```
int pe_UnconsUInt64 (
    OSCTXT * pctxt,
    OSUINT64 value )
```

This function encodes an unconstrained unsigned 64-bit integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded, represented as unsigned 64-bit integer.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.54 pe_UnconsUnsigned()

```
int pe_UnconsUnsigned (
    OSCTXT * pctxt,
    OSUINT32 value )
```

This function encodes an unconstrained unsigned integer.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.55 pe_UniversalString()

```
int pe_UniversalString (
    OSCTXT * pctxt,
    ASN1UniversalString value,
    Asn132BitCharSet * permCharSet )
```

This function will encode a variable of the ASN.1 Universal character string. This differs from the encode routines for the character strings previously described in that the Universal string type is based on 32-bit characters. A 32-bit character string is modeled using an array of unsigned integers.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	Character string to be encoded. The structure includes a count field containing all valid characters in the set as well as the aligned and unaligned bit counts required to encode the characters.
<i>permCharSet</i>	A pointer to the constraining character set. This contains an array containing all valid characters in the set as well as the aligned and the unaligned bit counts required to encode the characters.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.56 pe_VarWidthCharString()

```
int pe_VarWidthCharString (
    OSCTXT * pctxt,
    const char * value )
```

This function will encode a ASN.1 character string.

Parameters

<i>pctxt</i>	A pointer to a context structure. The provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>value</i>	A pointer to a character string containing the value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.57 pe_YearInt()

```
int pe_YearInt (
    OSCTXT * pctxt,
    OSINT32 value )
```

This function will encode an ISO 8601 YEAR type.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>value</i>	Value to be encoded.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.5.3.58 uperEncConstrString()

```
int uperEncConstrString (
    OSCTXT * pctxt,
    const char * string,
    const char * charSet,
    OSUINT32 nbits,
    OSUINT32 canSetBits )
```

This function encodes a constrained string value. This version supports unaligned PER only. It allows all of the required permitted alphabet constraint parameters to be passed in as arguments.

Parameters

<i>pctxt</i>	Pointer to context block structure.
<i>string</i>	Pointer to string to be encoded.
<i>charSet</i>	String containing permitted alphabet character set. Can be null if no character set was specified.
<i>nbits</i>	Number of bits in a character set character (unaligned).
<i>canSetBits</i>	Number of bits in a character from the canonical set representing this string.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.6 PER C Utility Functions

Macros

- #define **pd_setp**(pctxt, bufaddr, bufsiz, aligned) **pu_setBuffer**(pctxt, bufaddr, bufsiz, aligned)
- #define **pe_setp**(pctxt, bufaddr, bufsiz, aligned) **pu_setBuffer**(pctxt, bufaddr, bufsiz, aligned)

Functions

- int **pu_addSizeConstraint** (OSCTXT *pctxt, const Asn1SizeCnst *pSize)
- OSBOOL **pu_alignCharStr** (OSCTXT *pctxt, OSUINT32 len, OSUINT32 nbits, Asn1SizeCnst *pSize)
- OSUINT32 **pu_bitcnt** (OSUINT32 value)
- Asn1SizeCnst * **pu_checkSize** (Asn1SizeCnst *pSizeList, OSUINT32 value, OSBOOL *pExtendable)
- int **pu_checkSizeExt** (Asn1SizeCnst *pSizeCnst, OSSIZE value, OSBOOL *pExtendable, Asn1SizeValueRange *pSizeRange, OSBOOL *pExtSize)
- void **pu_freeContext** (OSCTXT *pctxt)
- size_t **pu_getMaskAndIndex** (size_t bitOffset, unsigned char *pMask)
- size_t **pu_getMsgLen** (OSCTXT *pctxt)
- size_t **pu_getMsgLenBits** (OSCTXT *pctxt)
- void **pu_hexdump** (OSCTXT *pctxt)
- int **pu_setBuffer** (OSCTXT *pctxt, OSOCTET *bufaddr, size_t bufsiz, OSBOOL aligned)
- int **pe_resetBuffer** (OSCTXT *pctxt)
- int **pu_initContext** (OSCTXT *pctxt, OSOCTET *bufaddr, OSUINT32 bufsiz, OSBOOL aligned)
- int **pu_initContextBuffer** (OSCTXT *pTarget, OSCTXT *pSource)
- const char * **pu_getFullName** (OSCTXT *pctxt, const char *suffix)
- void **pu_insLenField** (OSCTXT *pctxt)
- OSBOOL **pu_isFixedSize** (const Asn1SizeCnst *pSizeList)
- OSCTXT * **pu_newContext** (OSOCTET *bufaddr, OSUINT32 bufsiz, OSBOOL aligned)
- PERField * **pu_newField** (OSCTXT *pctxt, const char *nameSuffix)
- void **pu_initFieldList** (OSCTXT *pctxt, OSINT16 bitOffset)
- void **pu_initRtxDiagBitFieldList** (OSCTXT *pctxt, OSINT16 bitOffset)
- void **pu_popName** (OSCTXT *pctxt)
- void **pu_pushElemName** (OSCTXT *pctxt, int idx)
- void **pu_pushName** (OSCTXT *pctxt, const char *name)
- void **pu_setCharSet** (Asn1CharSet *pCharSet, const char *permSet)
- int **pu_set16BitCharSet** (OSCTXT *pctxt, Asn116BitCharSet *pCharSet, Asn116BitCharSet *pAlphabet)
- void **pu_set16BitCharSetFromRange** (Asn116BitCharSet *pCharSet, OSUINT16 firstChar, OSUINT16 lastChar)
- void **pu_setFldBitCount** (OSCTXT *pctxt)
- void **pu_setFldBitOffset** (OSCTXT *pctxt)
- void **pu_setFldListFromCtxt** (OSCTXT *pctxt, OSCTXT *srcctx)
- void **pu_setOpenTypeFldList** (OSCTXT *pctxt, OSRTSList *plist)
- void **pu_setRtxDiagOpenTypeFldList** (OSRTDiagBitFieldList *pMainBFList, OSRTDiagBitFieldList *pOpenTypeBFList)
- OSBOOL **pu_setTrace** (OSCTXT *pCtxt, OSBOOL value)
- void **pu_setAligned** (OSCTXT *pctxt, OSBOOL value)
- void **pu_deleteFieldList** (OSCTXT *pctxt)
- OSBOOL **pu_BitAndOctetStringAlignmentTest** (const Asn1SizeCnst *pSizeCnst, OSSIZE itemCount, OSBOOL bitStrFlag)
- void **pu_bindump** (OSCTXT *pctxt, const char *varname)

- void **pu_dumpField** (OSCTXT *pctx, [PERField](#) *pField, const char *varname, size_t nextBitOffset, [BinDump](#), [Buffer](#) *pbuf)
- int [pu_set32BitCharSet](#) (OSCTXT *pctx, Asn132BitCharSet *pCharSet, Asn132BitCharSet *pAlphabet)
- void [pu_set32BitCharSetFromRange](#) (Asn132BitCharSet *pCharSet, OSUINT32 firstChar, OSUINT32 lastChar)
- int [pu_GetLibVersion](#) (OSVOIDARG)
- const char * [pu_GetLibInfo](#) (OSVOIDARG)

6.6.1 Detailed Description

The PER utility functions are common routines used by both the PER encode and decode functions.

6.6.2 Function Documentation

6.6.2.1 pe_resetBuffer()

```
int pe_resetBuffer (
    OSCTXT * pctx )
```

This function resets the PER encode buffer to allow a new message to be encoded into it.

Parameters

<i>pctx</i>	Pointer to a context structure.
-------------	---------------------------------

Returns

Status of operation: 0 if successful, or a negative value if an error occurs.

6.6.2.2 pu_addSizeConstraint()

```
int pu_addSizeConstraint (
    OSCTXT * pctx,
    const Asn1SizeCnst * pSize )
```

This function is used to add size to a context variable.

Parameters

<i>pctx</i>	A pointer to a context structure. The referenced size constraint is added to this structure for use by a subsequent encode or decode function.
<i>pSize</i>	A pointer to the size constraint to add the context variable.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.6.2.3 pu_bindump()

```
void pu_bindump (
    OSCTXT * pctxt,
    const char * varname )
```

This function provides a detailed binary dump of the contents of the buffer currently specified in the given context. The list of fields dumped by this function was previously built up within the context using calls `pu_newField`, `pu_pushName`, and `pu_popName`. These calls are built into both compiler-generated and low-level PER encode/decode functions to trace the actual bit encoding of a given construct.

Parameters

<i>pctxt</i>	A pointer to a context structure. The contents of the encode or decode buffer that was specified in the call to <code>pu_initContext</code> or <code>pu_newContext</code> is dumped.
<i>varname</i>	The name of the top-level variable name of the structure being dumped.

Referenced by `ASN1PERMessageBuffer::binDump()`.

6.6.2.4 pu_checkSizeExt()

```
int pu_checkSizeExt (
    Asn1SizeCnst * pSizeCnst,
    OSSIZE value,
    OSBOOL * pExtendable,
    Asn1SizeValueRange * pSizeRange,
    OSBOOL * pExtSize )
```

This function checks if the given value falls within the root or extension part of the given size constraint.

Parameters

<i>pSizeCnst</i>	A pointer to a size constraint structure.
<i>value</i>	The value to be checked.
<i>pExtendable</i>	Pointer to boolean indicating if size constraint is extensible.
<i>pSizeRange</i>	Size range which value falls within.
<i>pExtSize</i>	Indicates if the size range is an extension range.

Returns

Status of the operation. 0 = success or RTERR_CONSVIO if size if not within the constraint bounds.

6.6.2.5 pu_freeContext()

```
void pu_freeContext (
    OSCTXT * pctx )
```

This function releases all dynamic memory associated with a context. This function should be called even if the referenced context variable is not dynamic. The reason is because it frees memory allocated within the context as well as the context structure (it will only try to free the context structure if it detects that it was previously allocated using the pu_newContext function).

Parameters

<i>pctx</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
-------------	--

6.6.2.6 pu_GetLibInfo()

```
const char* pu_GetLibInfo (
    OSVOIDARG )
```

Returns information string describing the library. The string contains name of library, its version and flags used for building the library.

Returns

Information string

6.6.2.7 pu_GetLibVersion()

```
int pu_GetLibVersion (
    OSVOIDARG )
```

Returns numeric version of run-time library. The format of version is as follows: MmP, where: M - major version number; m - minor version number; p - patch release number. For example, the value 581 means the version 5.81.

Returns

Version of run-time library in numeric format.

6.6.2.8 pu_getMsgLen()

```
size_t pu_getMsgLen (
    OSCTXT * pctxt )
```

This function will return the number of bytes in a PER message. This function is called after a compiler generated encode function is called to get the byte count of the encoded component.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

See also

[pu_getMsgLenBits](#)

Returns

Length (in bytes) of encoded message content.

Referenced by ASN1PERMessageBuffer::getMsgLen().

6.6.2.9 pu_getMsgLenBits()

```
size_t pu_getMsgLenBits (
    OSCTXT * pctxt )
```

This function will return the number of bits in a PER message. This function is called after a compiler generated encode function is called to get the bit count of the encoded component.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

See also

[pu_getMsgLen](#)

Returns

Length (in bits) of encoded message content.

6.6.2.10 pu_hexdump()

```
void pu_hexdump (
    OSCTXT * pctxt )
```

This function provides a standard hexadecimal dump of the contents of the buffer currently specified in the given context.

Parameters

<i>pctxt</i>	Pointer to a context structure. The contents of the encode or decode buffer that was specified in the call to pu_initContext or pu_newContext is dumped.
--------------	--

Referenced by ASN1PERMessageBuffer::hexDump().

6.6.2.11 pu_initContext()

```
int pu_initContext (
    OSCTXT * pctxt,
    OSOCTET * bufaddr,
    OSUINT32 bufsiz,
    OSBOOL aligned )
```

This function is used to initialize a pre-allocated OSCTXT structure. This can be an OSCTXT variable declared on the stack or a pointer to an OSCTXT structure that was previously allocated. This function sets all internal variables within the structure to their initial values.

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
<i>bufaddr</i>	For encoding, this is the address of a buffer to receive the encoded PER message (note: this is optional, if specified to NULL a dynamic buffer will be allocated). For decoding, this is that address of the buffer that contains the PER message to be decoded.
<i>bufsiz</i>	For encoding, this is the size of the encoded buffer (note: this is optional, if the bufaddr argument is specified to NULL, then dynamic encoding is in effect and the buffer size is indefinite). For decoding, this is the length (in octets) of the PER message to be decoded.
<i>aligned</i>	A Boolean value specifying whether aligned or unaligned encoding should be performed.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.6.2.12 pu_initContextBuffer()

```
int pu_initContextBuffer (
    OSCTXT * pTarget,
    OSCTXT * pSource )
```

This function is used to initialize the buffer of an OSCTXT structure with buffer data from a second context structure. This function copies the buffer information from the source context buffer to the destination structure. The non-buffer related fields in the context remain untouched.

Parameters

<i>pTarget</i>	A pointer to the target context structure. Buffer information within this structure is updated with data from the source context.
<i>pSource</i>	A pointer to the source context structure. Buffer information from the source context structure is copied to the target structure.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

6.6.2.13 pu_initFieldList()

```
void pu_initFieldList (
    OSCTXT * pctxt,
    OSINT16 bitOffset )
```

This function initializes a new bit field list for diagnostics tracing. It is now deprecated and has been replaced by function pu_initRtxDiagBitFieldList which works with the common context bit tracing list structure rather than the ASN.1 PER-specific list.

Parameters

<i>pctxt</i>	A pointer to a context structure.
<i>bitOffset</i>	Current bit offset.

6.6.2.14 pu_initRtxDiagBitFieldList()

```
void pu_initRtxDiagBitFieldList (
    OSCTXT * pctxt,
    OSINT16 bitOffset )
```

This function initializes a new bit field list for diagnostics tracing. It is used in code to encode or decode table-constrained open type data to break down the patterns within the containers.

Parameters

<i>pctxt</i>	A pointer to a context structure.
<i>bitOffset</i>	Current bit offset.

6.6.2.15 pu_insLenField()

```
void pu_insLenField (  
    OSCTXT * pctxt )
```

Parameters

<i>pctxt</i>	A pointer to a context structure. This provides a storage area for the function to store all working variables that must be maintained between function calls.
--------------	--

6.6.2.16 pu_isFixedSize()

```
OSBOOL pu_isFixedSize (  
    const Asn1SizeCnst * pSizeList )
```

This function determines if a size constraint is fixed (i.e allows only one value such as SIZE (2))

Parameters

<i>pSizeList</i>	A pointer to a size constraint structure.
------------------	---

Returns

TRUE if size constraint is fixed.

6.6.2.17 pu_newContext()

```
OSCTXT* pu_newContext (  
    OSOCTET * bufaddr,
```

```

    OSUINT32 bufsiz,
    OSBOOL aligned )

```

This function is similar to the `pu_initContext` function in that it initializes a context variable. The difference is that this function allocates a new structure and then initializes it. It is equivalent to calling `malloc` to allocate a context structure and then calling `pu_initContext` to initialize it.

Parameters

<i>bufaddr</i>	For encoding, this is the address of a buffer to receive the encoded PER message (note: this is optional, if specified as NULL a dynamic buffer will be allocated). For decoding, this is that address of the buffer that contains the PER message to be decoded.
<i>bufsiz</i>	For encoding, this is the size of the encoded buffer (note: this is optional, if the <i>bufaddr</i> argument is specified to NULL, then dynamic encoding is in effect and the buffer size is indefinite). For decoding, this is the length (in octets) of the PER message to be decoded.
<i>aligned</i>	A Boolean value specifying whether aligned or unaligned encoding should be performed.

Returns

A pointer to OSCTXT structure to receive the allocated structure. NULL is returned if any error occurs in allocating or initializing the context.

6.6.2.18 pu_newField()

```

PERField* pu_newField (
    OSCTXT * pctxt,
    const char * nameSuffix )

```

This function creates a new bit field in a diagnostics bit trace field list. It is now deprecated and has been replaced by the common function `rtxDiagNewBitField`.

Parameters

<i>pctxt</i>	A pointer to a context structure.
<i>nameSuffix</i>	Suffix to be append to current element name.

6.6.2.19 pu_popName()

```

void pu_popName (
    OSCTXT * pctxt )

```

Pop an element name from the diagnostics bit trace stack. This function is now deprecated and has been replaced with the common function `rtxCtxtPopElemName` which is used maintain a name stack for other purposes as well (for example, error reporting).

Parameters

<i>pctxt</i>	A pointer to a context structure.
--------------	-----------------------------------

6.6.2.20 pu_pushElemName()

```
void pu_pushElemName (
    OSCTXT * pctxt,
    int idx )
```

Push an array element on the diagnostics bit trace stack. This function is now deprecated and has been replaced with the common function `rtxCtxtPushArrayElemName` which is used maintain a name stack for other purposes as well (for example, error reporting).

Parameters

<i>pctxt</i>	A pointer to a context structure.
<i>idx</i>	The location to insert the element.

6.6.2.21 pu_pushName()

```
void pu_pushName (
    OSCTXT * pctxt,
    const char * name )
```

Push an element on the diagnostics bit trace stack. This function is now deprecated and has been replaced with the common function `rtxCtxtPushElemName` which is used maintain a name stack for other purposes as well (for example, error reporting).

Parameters

<i>pctxt</i>	A pointer to a context structure.
<i>name</i>	A pointer to the element to add to the stack.

6.6.2.22 pu_set16BitCharSet()

```
int pu_set16BitCharSet (
    OSCTXT * pctxt,
```



```

Asn116BitCharSet * pCharSet,
Asn116BitCharSet * pAlphabet )

```

This function sets a permitted alphabet character set for 16-bit character strings. This is the resulting set of character when the character associated with a 16-bit string type is merged with a permitted alphabet constraint.

Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>pCharSet</i>	Pointer to a character set structure describing the character set currently associated with the character string type. The resulting character set structure after being merged with the permSet parameter.
<i>pAlphabet</i>	Pointer to a structure describing the 16-bit permitted alphabet.

Returns

Status value of zero if successful, or a negative value if failure.

6.6.2.23 pu_set16BitCharSetFromRange()

```

void pu_set16BitCharSetFromRange (
    Asn116BitCharSet * pCharSet,
    OSUINT16 firstChar,
    OSUINT16 lastChar )

```

This function sets a permitted alphabet character set for 16-bit character strings from a sequential range of characters (for example, 'A'..'Z').

Parameters

<i>pCharSet</i>	Pointer to a character set structure describing the character set currently associated with the character string type. The resulting character set structure after being merged with the permSet parameter.
<i>firstChar</i>	The first character in the range.
<i>lastChar</i>	The last character in the range.

6.6.2.24 pu_set32BitCharSet()

```

int pu_set32BitCharSet (
    OSCTXT * pctxt,
    Asn132BitCharSet * pCharSet,
    Asn132BitCharSet * pAlphabet )

```

This function sets a permitted alphabet character set for 32-bit character strings. This is the resulting set of character when the character associated with a 16-bit string type is merged with a permitted alphabet constraint.

Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>pCharSet</i>	Pointer to a character set structure describing the character set currently associated with the character string type. The resulting character set structure after being merged with the permSet parameter.
<i>pAlphabet</i>	Pointer to a structure describing the 32-bit permitted alphabet.

Returns

Status value of zero if successful, or a negative value if failure.

6.6.2.25 pu_set32BitCharSetFromRange()

```
void pu_set32BitCharSetFromRange (
    Asn132BitCharSet * pCharSet,
    OSUINT32 firstChar,
    OSUINT32 lastChar )
```

This function sets a permitted alphabet character set for 32-bit character strings from a sequential range of characters (for example, 'A'..'Z').

Parameters

<i>pCharSet</i>	Pointer to a character set structure describing the character set currently associated with the character string type. The resulting character set structure after being merged with the permSet parameter.
<i>firstChar</i>	The first character in the range.
<i>lastChar</i>	The last character in the range.

6.6.2.26 pu_setBuffer()

```
int pu_setBuffer (
    OSCTXT * pctxt,
    OSOCTET * bufaddr,
    size_t bufsiz,
    OSBOOL aligned )
```

This function is used to set the buffer pointer for PER encoding or decoding. For encoding, the buffer would either be a static byte array in memory to receive the encoded message or, if bufaddr was set to NULL, would indicate encoding is to be done to a dynamic buffer. For decoding, a buffer in memory would be specified which contains the message to be decoded.

Parameters

<i>pctxt</i>	Pointer to a context structure.
<i>bufaddr</i>	Address of memory buffer. For encoding, this may be set to NULL to indicate a dynamic buffer is to be used.
<i>bufsiz</i>	Size, in byte, of static memory buffer.
<i>aligned</i>	Indicate if aligned (true) or unaligned (false) PER should be used for encoding or decoding.

6.6.2.27 pu_setCharSet()

```
void pu_setCharSet (
    Asn1CharSet * pCharSet,
    const char * permSet )
```

This function sets a permitted alphabet character set. This is the resulting set of characters when the character associated with a standard character string type is merged with a permitted alphabet constraint.

Parameters

<i>pCharSet</i>	A pointer to a character set structure describing the character set currently associated with the character string type. The resulting character set structure after being merged with the permSet parameter.
<i>permSet</i>	A null-terminated string of permitted characters.

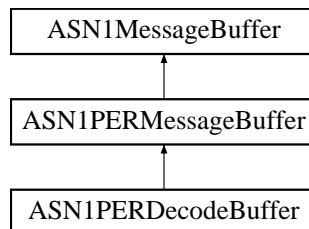
Chapter 7

Class Documentation

7.1 ASN1PERDecodeBuffer Class Reference

```
#include <asn1PerCppTypes.h>
```

Inheritance diagram for ASN1PERDecodeBuffer:



Public Member Functions

- [ASN1PERDecodeBuffer](#) (OSBOOL aligned)
- [ASN1PERDecodeBuffer](#) (const OSOCTET *pMsgBuf, size_t msgBufLen, OSBOOL aligned)
- [ASN1PERDecodeBuffer](#) (const OSOCTET *pMsgBuf, size_t msgBufLen, OSBOOL aligned, OSRTContext *pContext)
- EXTPERMETHOD [ASN1PERDecodeBuffer](#) (OSRTInputStream &istream, OSBOOL aligned)
- EXTPERMETHOD [ASN1PERDecodeBuffer](#) (const char *filePath, OSBOOL aligned)
- int [byteAlign](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)
- EXTPERMETHOD int [peekByte](#) (OSOCTET &ub)
- EXTPERMETHOD int [readBinaryFile](#) (const char *filePath)
- EXTPERMETHOD int [readBytes](#) (OSOCTET *buffer, size_t bufsize, size_t nbytes)

Additional Inherited Members

7.1.1 Detailed Description

The [ASN1PERDecodeBuffer](#) class is derived from the [ASN1PERMessageBuffer](#) base class. It contains variables and methods specific to decoding ASN.1 PER messages. It is used to manage the input buffer containing the ASN.1 message to be decoded. This class has 3 overloaded constructors.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 [ASN1PERDecodeBuffer\(\)](#) [1/5]

```
ASN1PERDecodeBuffer::ASN1PERDecodeBuffer (  
    OSBOOL aligned ) [inline]
```

This is a default constructor. Use `getStatus()` method to determine has error occurred during the initialization or not.

Parameters

<i>aligned</i>	Flag indicating if the message was encoded using aligned (TRUE)* or unaligned (FALSE) encoding.
----------------	---

7.1.2.2 [ASN1PERDecodeBuffer\(\)](#) [2/5]

```
ASN1PERDecodeBuffer::ASN1PERDecodeBuffer (  
    const OSOCTET * pMsgBuf,  
    size_t msgBufLen,  
    OSBOOL aligned ) [inline]
```

This constructor is used to describe the message to be decoded. Use `getStatus()` method to determine has error occurred during the initialization or not.

Parameters

<i>pMsgBuf</i>	A pointer to the message to be decoded.
<i>msgBufLen</i>	Length of the message buffer.
<i>aligned</i>	Flag indicating if the message was encoded using aligned (TRUE) * or unaligned (FALSE) encoding.

7.1.2.3 ASN1PERDecodeBuffer() [3/5]

```
ASN1PERDecodeBuffer::ASN1PERDecodeBuffer (
    const OSOCTET * pMsgBuf,
    size_t msgBufLen,
    OSBOOL aligned,
    OSRTContext * pContext ) [inline]
```

This constructor is used to describe the message to be decoded. Use `getStatus()` method to determine has error occurred during the initialization or not.

Parameters

<i>pMsgBuf</i>	A pointer to the message to be decoded.
<i>msgBufLen</i>	Length of the message buffer.
<i>aligned</i>	Flag indicating if the message was encoded using aligned (TRUE) * or unaligned (FALSE) encoding.
<i>pContext</i>	A pointer to an OSRTContext structure created by the user.

7.1.2.4 ASN1PERDecodeBuffer() [4/5]

```
EXTPERMETHOD ASN1PERDecodeBuffer::ASN1PERDecodeBuffer (
    OSRTInputStream & istream,
    OSBOOL aligned )
```

This version of the [ASN1PERDecodeBuffer](#) constructor takes a reference to an input stream object and an aligned flag argument (stream decoding version).

Parameters

<i>istream</i>	A reference to an input stream object.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

7.1.2.5 ASN1PERDecodeBuffer() [5/5]

```
EXTPERMETHOD ASN1PERDecodeBuffer::ASN1PERDecodeBuffer (
    const char * filePath,
    OSBOOL aligned )
```

This constructor takes a pointer to the path of a file containing a binary PER message to be decoded.

Parameters

<i>filePath</i>	Complete file path and name of file to read.
<i>aligned</i>	Flag indicating if the message was encoded using aligned (TRUE) * or unaligned (FALSE) encoding.

7.1.3 Member Function Documentation

7.1.3.1 byteAlign()

```
int ASN1PERDecodeBuffer::byteAlign ( ) [inline]
```

This method aligns the input buffer or stream to the next octet boundary.

Returns

Status of operation: 0 = success, negative value if error occurred.

7.1.3.2 isA()

```
virtual OSBOOL ASN1PERDecodeBuffer::isA (
    Type bufferType ) [inline], [virtual]
```

This method checks the type of the message buffer.

Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. The only possible value for this class is PERDecode.
-------------------	--

Returns

Boolean result of the match operation. True if this is the class corresponding to the identifier argument.

7.1.3.3 peekByte()

```
EXTERNMETHOD int ASN1PERDecodeBuffer::peekByte (
    OSOCKET & ub )
```

This method is used to peek at the next available byte in the decode buffer/stream without advancing the cursor.

Parameters

<i>ub</i>	Single byte buffer to receive peeked byte.
-----------	--

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

7.1.3.4 readBinaryFile()

```
EXTPERMETHOD int ASN1PERDecodeBuffer::readBinaryFile (  
    const char * filePath )
```

This method reads the file into the buffer to decode.

Parameters

<i>filePath</i>	The zero-terminated string containing the path to the file.
-----------------	---

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

7.1.3.5 readBytes()

```
EXTPERMETHOD int ASN1PERDecodeBuffer::readBytes (  
    OSOCKET * buffer,  
    size_t bufsize,  
    size_t nbytes )
```

This method is used to read the given number of bytes from the underlying buffer/stream into the given buffer.

Parameters

<i>buffer</i>	Buffer into which data should be read.
<i>bufsize</i>	Size of the buffer
<i>nbytes</i>	Number of bytes to read. Must be <= bufsize.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

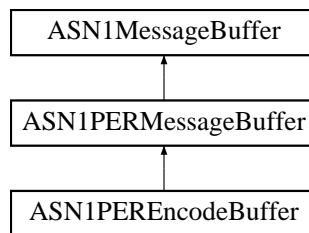
The documentation for this class was generated from the following file:

- [asn1PerCppTypes.h](#)

7.2 ASN1PEREncodeBuffer Class Reference

```
#include <asn1PerCppTypes.h>
```

Inheritance diagram for ASN1PEREncodeBuffer:



Public Member Functions

- [ASN1PEREncodeBuffer](#) (OSBOOL aligned)
- [ASN1PEREncodeBuffer](#) (OSOCKET *pMsgBuf, size_t msgBufLen, OSBOOL aligned)
- [ASN1PEREncodeBuffer](#) (OSOCKET *pMsgBuf, size_t msgBufLen, OSBOOL aligned, OSRTContext *pContext)
- [ASN1PEREncodeBuffer](#) (OSRTOutputStream &ostream, OSBOOL aligned)
- int [byteAlign](#) ()
- int [encodeBit](#) (OSBOOL value)
- int [encodeBits](#) (const OSOCKET *pvalue, size_t nbits, OSUINT32 bitOffset=0)
- size_t [getMsgBitCnt](#) ()
- virtual EXTPERMETHOD OSOCKET * [getMsgCopy](#) ()
- virtual EXTPERMETHOD const OSOCKET * [getMsgPtr](#) ()
- EXTPERMETHOD int [init](#) ()
- virtual OSBOOL [isA](#) (Type bufferType)
- int [GetMsgBitCnt](#) ()

Additional Inherited Members

7.2.1 Detailed Description

The [ASN1PEREncodeBuffer](#) class is derived from the [ASN1PERMessageBuffer](#) base class. It contains variables and methods specific to encoding ASN.1 messages. It is used to manage the buffer into which an ASN.1 PER message is to be encoded.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ASN1PEREncodeBuffer() [1/4]

```
ASN1PEREncodeBuffer::ASN1PEREncodeBuffer (
    OSBOOL aligned ) [inline]
```

This version of the [ASN1PEREncodeBuffer](#) constructor that takes one argument, *aligned* flag. It creates a dynamic memory buffer into which a PER message is encoded.

Parameters

<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.
----------------	---

7.2.2.2 ASN1PEREncodeBuffer() [2/4]

```
ASN1PEREncodeBuffer::ASN1PEREncodeBuffer (
    OSOCKET * pMsgBuf,
    size_t msgBufLen,
    OSBOOL aligned ) [inline]
```

This version of the [ASN1PEREncodeBuffer](#) constructor takes a message buffer and size argument and an *aligned* flag argument (static encoding version).

Parameters

<i>pMsgBuf</i>	A pointer to a fixed-size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

7.2.2.3 ASN1PEREncodeBuffer() [3/4]

```
ASN1PEREncodeBuffer::ASN1PEREncodeBuffer (
    OSOCKET * pMsgBuf,
    size_t msgBufLen,
    OSBOOL aligned,
    OSRTContext * pContext ) [inline]
```

This version of the [ASN1PEREncodeBuffer](#) constructor takes a message buffer and size argument and an *aligned* flag argument (static encoding version) as well as a pointer to an existing context object.

Parameters

<i>pMsgBuf</i>	A pointer to a fixed-size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.
<i>pContext</i>	A pointer to an OSRTContext structure created by the user.

7.2.2.4 ASN1PEREncodeBuffer() [4/4]

```
ASN1PEREncodeBuffer::ASN1PEREncodeBuffer (
    OSRTOutputStream & ostream,
    OSBOOL aligned )
```

This version of the [ASN1PEREncodeBuffer](#) constructor takes a reference to an output stream object and an aligned flag argument (stream encoding version).

Parameters

<i>ostream</i>	A reference to an output stream object.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

7.2.3 Member Function Documentation

7.2.3.1 byteAlign()

```
int ASN1PEREncodeBuffer::byteAlign ( ) [inline]
```

This method aligns the output buffer or stream to the next octet boundary.

Returns

Status of operation: 0 = success, negative value if error occurred.

References [pe_byte_align\(\)](#).

7.2.3.2 encodeBit()

```
int ASN1PEREncodeBuffer::encodeBit (
    OSBOOL value ) [inline]
```

This method writes a single encoded bit value to the output buffer or stream.

Parameters

<i>value</i>	Boolean value of bit to be written.
--------------	-------------------------------------

Returns

Status of operation: 0 = success, negative value if error occurred.

7.2.3.3 encodeBits()

```
int ASN1PEREncodeBuffer::encodeBits (
    const OSOCTET * pvalue,
    size_t nbits,
    OSUINT32 bitOffset = 0 ) [inline]
```

This method writes the given number of bits from the byte array to the output buffer or stream starting from the given bit offset.

Parameters

<i>pvalue</i>	Pointer to byte array containing data to be encoded.
<i>nbits</i>	Number of bits to copy from byte array to encode buffer.
<i>bitOffset</i>	Starting bit offset from which bits are to be copied.

Returns

Status of operation: 0 = success, negative value if error occurred.

7.2.3.4 getMsgBitCnt()

```
size_t ASN1PEREncodeBuffer::getMsgBitCnt ( ) [inline]
```

This method returns the length (in bits) of the encoded message.

Returns

Length(in bits)of encoded message

References `pe_GetMsgBitCnt()`.

7.2.3.5 getMsgCopy()

```
virtual EXTPERMETHOD OSOCTET* ASN1PEREncodeBuffer::getMsgCopy ( ) [virtual]
```

This method returns a copy of the current encoded message. Memory is allocated for the message using the 'new' operation. It is the user's responsibility to free the memory using 'delete'.

Returns

Pointer to copy of encoded message. It is the user's responsibility to release the memory using the 'delete' operator (i.e., delete [] ptr;)

7.2.3.6 getMsgPtr()

```
virtual EXTPERMETHOD const OSOCTET* ASN1PEREncodeBuffer::getMsgPtr ( ) [virtual]
```

This method returns the internal pointer to the current encoded message.

Returns

Pointer to encoded message.

7.2.3.7 init()

```
EXTPERMETHOD int ASN1PEREncodeBuffer::init ( )
```

This method reinitializes the encode buffer pointer to allow a new message to be encoded. This makes it possible to reuse one message buffer object in a loop to encode multiple messages. After this method is called, any previously encoded message in the buffer will be overwritten on the next encode call.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

7.2.3.8 isA()

```
virtual OSBOOL ASN1PEREncodeBuffer::isA (
    Type bufferType ) [inline], [virtual]
```

This method checks the type of the message buffer.

Parameters

<i>bufferType</i>	Enumerated identifier specifying a derived class. The only possible value for this class is PEREncode.
-------------------	--

Returns

Boolean result of the match operation. True if this is the class corresponding to the identifier argument.

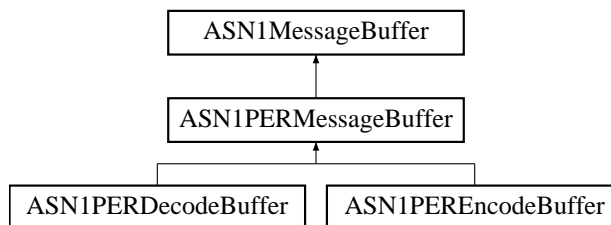
The documentation for this class was generated from the following file:

- [asn1PerCppTypes.h](#)

7.3 ASN1PERMessageBuffer Class Reference

```
#include <asn1PerCppTypes.h>
```

Inheritance diagram for ASN1PERMessageBuffer:



Public Member Functions

- void `binDump` (const char *varname)
- void `hexDump` ()
- virtual size_t `getMsgLen` ()
- OSBOOL `isAligned` ()
- void `setTrace` (OSBOOL value)
- EXTPERMETHOD int `setBuffer` (const OSOCTET *pMsgBuf, size_t msgBufLen)
- void **BinDump** (const char *varname)
- void **HexDump** ()
- int **GetMsgLen** ()
- void **SetTrace** (OSBOOL value)

Protected Member Functions

- EXTPERMETHOD `ASN1PERMessageBuffer` (Type bufferType, OSBOOL aligned)
- EXTPERMETHOD `ASN1PERMessageBuffer` (OSRTStream &stream, OSBOOL aligned)
- EXTPERMETHOD `ASN1PERMessageBuffer` (Type bufferType, OSOCTET *pMsgBuf, size_t msgBufLen, OS↔
BOOL aligned)
- EXTPERMETHOD `ASN1PERMessageBuffer` (Type bufferType, OSOCTET *pMsgBuf, size_t msgBufLen, OS↔
BOOL aligned, OSRTContext *pContext)

7.3.1 Detailed Description

The [ASN1PERMessageBuffer](#) class is derived from the [ASN1MessageBuffer](#) base class. It is the base class for the [ASN1PEREncodeBuffer](#) and [ASN1PERDecodeBuffer](#) derived classes. It contains variables and methods specific to encoding or decoding ASN.1 messages using the Packed Encoding Rules (PER). It is used to manage the buffer into which an ASN.1 message is to be encoded or decoded.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 [ASN1PERMessageBuffer\(\)](#) [1/4]

```
EXTPERMETHOD ASN1PERMessageBuffer::ASN1PERMessageBuffer (  
    Type bufferType,  
    OSBOOL aligned ) [protected]
```

This constructor does not set a PER input source. It is used by the derived encode buffer classes. Use the `getStatus()` method to determine if an error has occurred during initialization.

Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, PEREncode or PERDecode).
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

7.3.2.2 [ASN1PERMessageBuffer\(\)](#) [2/4]

```
EXTPERMETHOD ASN1PERMessageBuffer::ASN1PERMessageBuffer (  
    OSRTStream & stream,  
    OSBOOL aligned ) [protected]
```

This constructor associates a stream with a PER encode or decode buffer. It is used by the derived encode buffer classes to create a stream-based PER encoder or decoder.

Parameters

<i>stream</i>	Stream class reference.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

7.3.2.3 ASN1PERMessageBuffer() [3/4]

```
EXTPERMETHOD ASN1PERMessageBuffer::ASN1PERMessageBuffer (
    Type bufferType,
    OSOCKET * pMsgBuf,
    size_t msgBufLen,
    OSBOOL aligned ) [protected]
```

This constructor allows a memory buffer holding a binary PER message to be specified. Use the `getStatus()` method to determine if an error has occurred during initialization.

Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, PEREncode or PERDecode).
<i>pMsgBuf</i>	A pointer to a fixed size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

7.3.2.4 ASN1PERMessageBuffer() [4/4]

```
EXTPERMETHOD ASN1PERMessageBuffer::ASN1PERMessageBuffer (
    Type bufferType,
    OSOCKET * pMsgBuf,
    size_t msgBufLen,
    OSBOOL aligned,
    OSRTContext * pContext ) [protected]
```

This constructor allows a memory buffer holding a binary PER message to be specified. It also allows a pre-existing context to be associated with this buffer. Use the `getStatus()` method to determine if an error has occurred during initialization.

Parameters

<i>bufferType</i>	Type of message buffer that is being created (for example, PEREncode or PERDecode).
<i>pMsgBuf</i>	A pointer to a fixed size message buffer to receive the encoded message.
<i>msgBufLen</i>	Size of the fixed-size message buffer.
<i>aligned</i>	Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.
<i>pContext</i>	A pointer to an OSRTContext structure.

7.3.3 Member Function Documentation

7.3.3.1 binDump()

```
void ASN1PERMessageBuffer::binDump (
    const char * varname ) [inline]
```

This method outputs a binary dump of the current buffer contents to stdout.

Parameters

<i>varname</i>	char pointer to current buffer
----------------	--------------------------------

References pu_bindump().

7.3.3.2 getMsgLen()

```
virtual size_t ASN1PERMessageBuffer::getMsgLen ( ) [inline], [virtual]
```

This method returns the length of a previously encoded PER message.

Parameters

-	none
---	------

References pu_getMsgLen().

7.3.3.3 hexDump()

```
void ASN1PERMessageBuffer::hexDump ( ) [inline]
```

This method outputs a hexadecimal dump of the current buffer contents to stdout.

Parameters

-	none
---	------

References pu_hexdump().

7.3.3.4 isAligned()

```
OSBOOL ASN1PERMessageBuffer::isAligned ( ) [inline]
```

This method indicates if PER aligned encoding is in effect.

Parameters

-	none
---	------

Returns

Boolean result: true if aligned; false if unaligned.

7.3.3.5 setBuffer()

```
EXTPERMETHOD int ASN1PERMessageBuffer::setBuffer (
    const OSOCTET * pMsgBuf,
    size_t msgBufLen )
```

This method sets a buffer to receive the encoded message.

Parameters

<i>pMsgBuf</i>	A pointer to a memory buffer to use to encode a message. The buffer should be declared as an array of unsigned characters (OSOCTETs). This parameter can be set to NULL to specify dynamic encoding (i.e., the encode functions will dynamically allocate a buffer for the message).
<i>msgBufLen</i>	The length of the memory buffer in bytes. If pMsgBuf is NULL, this parameter specifies the initial size of the dynamic buffer; if 0 - the default size will be used.

Returns

Completion status of operation:

- 0 (0) = success,
- negative return value is error.

7.3.3.6 setTrace()

```
void ASN1PERMessageBuffer::setTrace (
    OSBOOL value ) [inline]
```

This method turns PER diagnostic tracing on or off.

This enables the collection of the bit statistics inside the PER library functions that can be displayed using the binDump method.

Parameters

<i>value</i>	Boolean value indicating whether tracing should be on (true) or off (false).
--------------	--

The documentation for this class was generated from the following file:

- [asn1PerCppType.h](#)

7.4 BinDumpBuffer Struct Reference

Public Attributes

- unsigned char **lb**
- unsigned char **lbn**
- char **fmtBitBuffer** [40]
- char **fmtHexBuffer** [10]
- char **fmtAscBuffer** [10]
- int **fmtBitCharIdx**
- int **fmtHexCharIdx**
- int **fmtAscCharIdx**

The documentation for this struct was generated from the following file:

- [asn1per.h](#)

7.5 PERField Struct Reference

Public Attributes

- const char * **name**
- size_t **bitOffset**
- size_t **numbits**
- OSRTSList * **openTypeFields**

The documentation for this struct was generated from the following file:

- [asn1per.h](#)

Chapter 8

File Documentation

8.1 asn1per.h File Reference

```
#include "rtsrc/asn1type.h"
#include "rtsrc/asn1CharSet.h"
#include "rtxsrc/rtxBitEncode.h"
#include "rtxsrc/rtxBitDecode.h"
#include "rtxsrc/rtxBuffer.h"
#include "rtxsrc/rtxDiagBitTrace.h"
```

Classes

- struct [PERField](#)
- struct [BinDumpBuffer](#)

Macros

- #define **ASN_K_EXTENUM** 999
- #define **MAX_BIGINTBYTES** (1024)
- #define **OSYEAR_BASIC** OSUINTCONST(0x8000000)
- #define **OSYEAR_PROLEPTIC** OSUINTCONST(0x4000000)
- #define **OSYEAR_NEGATIVE** OSUINTCONST(0x2000000)
- #define **OSYEAR_L(n)** ((OSUINT32)(n) << 28)
- #define **OSYEAR_MASK** (OSYEAR_BASIC|OSYEAR_PROLEPTIC|OSYEAR_NEGATIVE|OSYEAR_L(0xF))
- #define **OSANY** (OSYEAR_NEGATIVE|OSYEAR_L(5))
- #define **OSANY_MASK** (OSYEAR_NEGATIVE|OSYEAR_L(0xF))
- #define **OSCENTURY** 0x4000u
- #define **OSYEAR** 0x2000u
- #define **OSMONTH** 0x1000u
- #define **OSWEEK** 0x0800u
- #define **OSDAY** 0x0400u

- #define **OSHOURS** 0x0200u
- #define **OSMINUTES** 0x0100u
- #define **OSSECONDS** 0x0080u
- #define **OSUTC** 0x0040u
- #define **OSDIFF** 0x0020u
- #define **OSFRACTION** 0x000Fu
- #define **OSDURATION** 0x0010u
- #define **PU_SETCHARSET**(csetvar, canset, abits, ubits)
- #define **PU_INSLENFLD**(pctxt)
- #define **PU_NEWFIELD**(pctxt, suffix)
- #define **PU_PUSHNAME**(pctxt, name)
- #define **PU_PUSHELEMNAME**(pctxt, idx)
- #define **PU_POPNAME**(pctxt)
- #define **PU_SETBITOFFSET**(pctxt)
- #define **PU_SETBITCOUNT**(pctxt)
- #define **PU_SETOPENTYPEFLDLIST**(pMainBFList, pOpenTypeBFList)
- #define **EXTPERMETHOD**
- #define **EXTPERCLASS**
- #define **PD_BIT**(pctxt, pvalue) DEC_BIT(pctxt,pvalue)
- #define **PU_SETSIZECONSTRAINT**(pctxt, rootLower, rootUpper, extLower, extUpper)
- #define **PU_INITSIZECONSTRAINT**(pctxt) PU_SETSIZECONSTRAINT(pctxt,0,0,0,0)
- #define **PU_GETSIZECONSTRAINT**(pctxt, extbit)
- #define **PU_GETCTXTBITOFFSET**(pctxt) (((pctxt)->buffer.byteIndex * 8) + (8 - (pctxt)->buffer.bitOffset))
- #define **PU_GETPADBITS**(pctxt) (((pctxt)->buffer.bitOffset == 8) ? 0 : (pctxt)->buffer.bitOffset)
- #define **PU_SETCTXTBITOFFSET**(pctxt, _bitOffset)
- #define **PD_BYTE_ALIGN0**(pctxt)
- #define **PD_BYTE_ALIGN** PD_BYTE_ALIGN0
- #define **PD_CHECKSEQOFLEN**(pctxt, numElements, minElemBits)
- #define **pd_moveBitCursor**(pctxt, bitOffset) rtxMoveBitCursor(pctxt,bitOffset)
- #define **pe_bit**(pctxt, value) rtxEncBit(pctxt,value)
- #define **pe_bits**(pctxt, value, nbits) rtxEncBits(pctxt,value,nbits)
- #define **pe_CheckBuffer**(pctxt, nbytes) rtxCheckOutputBuffer(pctxt,nbytes)
- #define **pe_ConsInteger**(pctxt, value, lower, upper) **pe_ConsInt64**(pctxt, value, lower, upper)
- #define **pe_ConsUnsigned**(pctxt, value, lower, upper) **pe_ConsUInt64**(pctxt, value, lower, upper)
- #define **pe_ConsUnsignedSignedBound**(pcxt, value, lower, upper) **pe_ConsUInt64SignedBound**(pctxt, value, lower, upper)
- #define **pe_octets**(pctxt, pvalue, nbits) rtxEncBitsFromArray(pctxt,pvalue,nbits)
- #define **pe_SemiConsInteger**(pctxt, value, lower) **pe_SemiConsInt64**(pctxt, value, lower)
- #define **pe_SemiConsUnsigned**(pctxt, value, lower) **pe_SemiConsUInt64**(pctxt, value, lower)
- #define **pd_setp**(pctxt, bufaddr, bufsiz, aligned) **pu_setBuffer**(pctxt, bufaddr, bufsiz, aligned)
- #define **pe_setp**(pctxt, bufaddr, bufsiz, aligned) **pu_setBuffer**(pctxt, bufaddr, bufsiz, aligned)
- #define **pd_bit**(pctxt, pvalue) rtxDecBit(pctxt,pvalue)
- #define **pd_bits**(pctxt, pvalue, nbits) rtxDecBits(pctxt,pvalue,nbits)
- #define **pd_octets**(pctxt, pBuffer, bufsiz, nbits) rtxDecBitsToByteArray(pctxt,pBuffer,bufsiz,nbits)
- #define **pe_GeneralString**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)
- #define **pe_GraphicString**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)
- #define **pe_T61String**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)
- #define **pe_TeletexString**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)
- #define **pe_VideotexString**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)
- #define **pe_ObjectDescriptor**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)
- #define **pe_UTF8String**(pctxt, value, permCharSet) **pe_VarWidthCharString**(pctxt, value)

- #define **pe_IA5String**(pctxt, value, permCharSet) [pe_ConstrainedStringEx](#) (pctxt, value, permCharSet, 8, 7, 7)
- #define **pe_NumericString**(pctxt, value, permCharSet)
- #define **pe_PrintableString**(pctxt, value, permCharSet) [pe_ConstrainedStringEx](#) (pctxt, value, permCharSet, 8, 7, 7)
- #define **pe_VisibleString**(pctxt, value, permCharSet) [pe_ConstrainedStringEx](#) (pctxt, value, permCharSet, 8, 7, 7)
- #define **pe_ISO646String** pe_IA5String
- #define **pe_GeneralizedTime** pe_IA5String
- #define **pe_UTCTime** pe_GeneralizedTime
- #define **pd_GeneralString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_GraphicString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_VideotexString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_TeletexString**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_T61String**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_ObjectDescriptor**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_UTF8String**(pctxt, pvalue, permCharSet) [pd_VarWidthCharString](#) (pctxt, pvalue)
- #define **pd_IA5String**(pctxt, pvalue, permCharSet) [pd_ConstrainedStringEx](#) (pctxt, pvalue, permCharSet, 8, 7, 7)
- #define **pd_NumericString**(pctxt, pvalue, permCharSet)
- #define **pd_PrintableString**(pctxt, pvalue, permCharSet) [pd_ConstrainedStringEx](#) (pctxt, pvalue, permCharSet, 8, 7, 7)
- #define **pd_VisibleString**(pctxt, pvalue, permCharSet) [pd_ConstrainedStringEx](#) (pctxt, pvalue, permCharSet, 8, 7, 7)
- #define **pd_ISO646String** pd_IA5String
- #define **pd_GeneralizedTime** pd_IA5String
- #define **pd_UTCTime** pd_GeneralizedTime
- #define **pe_GetMsgLen** [pu_getMsgLen](#)
- #define **pe_ExpandBuffer**(pctxt, nbytes) [rtxExpandOutputBuffer](#)(pctxt, nbytes)
- #define **pd_AnyCentury**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSCENTURY)
- #define **pd_AnyCenturyInt**(pctxt, pvalue) [pd_UnconsInteger](#) (pctxt, pvalue)
- #define **pd_AnyDate**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pd_AnyYear**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR)
- #define **pd_AnyYearInt**(pctxt, pvalue) [pd_UnconsInteger](#) (pctxt, pvalue)
- #define **pd_AnyYearDay**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSDAY)
- #define **pd_AnyYearMonth**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSMONTH)
- #define **pd_AnyYearMonthDay**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pd_AnyYearWeek**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSWEEK)
- #define **pd_AnyYearWeekDay**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSANY|OSYEAR|OSWEEK|OSDAY)
- #define **pd_Century**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSCENTURY)
- #define **pd_CenturyInt**(pctxt, pvalue) [pd_ConsUInt8](#) (pctxt, pvalue, 0, 99)
- #define **pd_Date**(pctxt, string) [pd_DateStr](#) (pctxt, string, OSYEAR_BASIC|OSYEAR|OSMONTH|OSDAY);
- #define **pd_DateTime**(pctxt, string)
- #define **pd_DurationInterval**(pctxt, string) [pd_Duration](#) (pctxt, string, FALSE)
- #define **pd_DurationEndDateInterval**(pctxt, string, flags) [pd_Interval](#) (pctxt, string, FALSE, OSDURATION, flags)
- #define **pd_DurationEndTimeInterval**(pctxt, string, flags) [pd_Interval](#) (pctxt, string, FALSE, OSDURATION, flags)
- #define **pd_DurationEndDateTimeInterval**(pctxt, string, flags) [pd_Interval](#) (pctxt, string, FALSE, OSDURATION, flags)
- #define **pd_Hours**(pctxt, string) [pd_TimeStr](#) (pctxt, string, OSHOURS)
- #define **pd_HoursUtc**(pctxt, string) [pd_TimeStr](#) (pctxt, string, OSHOURS|OSUTC)

- #define **pd_HoursAndDiff**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSDIFF)
- #define **pd_HoursAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|(n))
- #define **pd_HoursUtcAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSUTC|(n))
- #define **pd_HoursAndDiffAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSDIFF|(n))
- #define **pd_Minutes**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES)
- #define **pd_MinutesUtc**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSUTC)
- #define **pd_MinutesAndDiff**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSDIFF)
- #define **pd_MinutesAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|(n))
- #define **pd_MinutesUtcAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSUTC|(n))
- #define **pd_MinutesAndDiffAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSDIFF|(n))
- #define **pd_RecStartEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pd_RecStartEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pd_RecStartEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pd_RecDurationInterval**(pctxt, string) **pd_Duration** (pctxt, string, TRUE)
- #define **pd_RecStartDateDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, OSDURATION)
- #define **pd_RecStartTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, OSDURATION)
- #define **pd_RecStartDateTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, flags, OSDURATION)
- #define **pd_RecDurationEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, OSDURATION, flags)
- #define **pd_RecDurationEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, OSDURATION, flags)
- #define **pd_RecDurationEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, TRUE, OSDURATION, flags)
- #define **pd_StartEndDateInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pd_StartEndTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pd_StartEndDateTimeInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, flags)
- #define **pd_StartDateDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, OSDURATION)
- #define **pd_StartTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, OSDURATION)
- #define **pd_StartDateTimeDurationInterval**(pctxt, string, flags) **pd_Interval** (pctxt, string, FALSE, flags, OSDURATION)
- #define **pd_TimeOfDay**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS)
- #define **pd_TimeOfDayUtc**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSUTC)
- #define **pd_TimeOfDayAndDiff**(pctxt, string) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSDIFF)
- #define **pd_TimeOfDayAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|(n))
- #define **pd_TimeOfDayUtcAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSUTC|(n))
- #define **pd_TimeOfDayAndDiffAndFraction**(pctxt, string, n) **pd_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSDIFF|(n))
- #define **pd_Year**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR)
- #define **pd_YearDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSDAY)
- #define **pd_YearMonth**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSMONTH)
- #define **pd_YearMonthDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSMONTH|OSDAY);

- #define **pd_YearWeek**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSWEEK)
- #define **pd_YearWeekDay**(pctxt, string) **pd_DateStr** (pctxt, string, OSYEAR|OSWEEK|OSDAY)
- #define **pe_AnyCentury**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSCENTURY)
- #define **pe_AnyCenturyInt**(pctxt, value) **pe_UnconsInteger** (pctxt, value)
- #define **pe_AnyDate**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pe_AnyYear**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR)
- #define **pe_AnyYearInt**(pctxt, value) **pe_UnconsInteger** (pctxt, value)
- #define **pe_AnyYearDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSDAY)
- #define **pe_AnyYearMonth**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH)
- #define **pe_AnyYearMonthDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSMONTH|OSDAY)
- #define **pe_AnyYearWeek**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSWEEK)
- #define **pe_AnyYearWeekDay**(pctxt, string) **pe_DateStr** (pctxt, string, OSANY|OSYEAR|OSWEEK|OSDAY)
- #define **pe_Century**(pctxt, string) **pe_DateStr** (pctxt, string, OSCENTURY)
- #define **pe_CenturyInt**(pctxt, value) **pe_ConsUnsigned** (pctxt, value, 0, 99)
- #define **pe_Date**(pctxt, string) **pe_DateStr** (pctxt, string, OSYEAR_BASIC|OSYEAR|OSMONTH|OSDAY)
- #define **pe_DateTime**(pctxt, string)
- #define **pe_DurationInterval**(pctxt, string) **pe_Duration** (pctxt, string, FALSE)
- #define **pe_DurationEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pe_DurationEndTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pe_DurationEndDateTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, FALSE, OSDURATION, flags)
- #define **pe_Hours**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS)
- #define **pe_HoursUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSUTC)
- #define **pe_Hours**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS)
- #define **pe_HoursUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSUTC)
- #define **pe_HoursAndDiff**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSDIFF)
- #define **pe_HoursAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|(n))
- #define **pe_HoursUtcAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSUTC|(n))
- #define **pe_HoursAndDiffAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSDIFF|(n))
- #define **pe_Minutes**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES)
- #define **pe_MinutesUtc**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSUTC)
- #define **pe_MinutesAndDiff**(pctxt, string) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OSDIFF)
- #define **pe_MinutesAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|(n))
- #define **pe_MinutesUtcAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUTES|OS←
UTC|(n))
- #define **pe_MinutesAndDiffAndFraction**(pctxt, string, n) **pe_TimeStr** (pctxt, string, OSHOURS|OSMINUT←
ES|OSDIFF|(n))
- #define **pe_RecStartEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pe_RecStartEndTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pe_RecStartEndDateTimeInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, flags)
- #define **pe_RecDurationInterval**(pctxt, string) **pe_Duration** (pctxt, string, TRUE)
- #define **pe_RecStartDateDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, OSDUR←
ATION)
- #define **pe_RecStartTimeDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, OSDU←
RATION)
- #define **pe_RecStartDateTimeDurationInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, flags, O←
SDURATION)
- #define **pe_RecDurationEndDateInterval**(pctxt, string, flags) **pe_Interval** (pctxt, string, TRUE, OSDURATION, flags)

- #define **pe_RecDurationEndTimeInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, TRUE, OSDURATION, flags)
- #define **pe_RecDurationEndDateTimeInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, TRUE, OSDURATION, flags)
- #define **pe_StartEndDateInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, FALSE, flags, flags)
- #define **pe_StartEndTimeInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, FALSE, flags, flags)
- #define **pe_StartEndDateTimeInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, FALSE, flags, flags)
- #define **pe_StartDateDurationInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, FALSE, flags, OSDURATION)
- #define **pe_StartTimeDurationInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, FALSE, flags, OSDURATION)
- #define **pe_StartDateTimeDurationInterval**(pctxt, string, flags) [pe_Interval](#) (pctxt, string, FALSE, flags, OSDURATION)
- #define **pe_TimeOfDay**(pctxt, string) [pe_TimeStr](#) (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS)
- #define **pe_TimeOfDayUtc**(pctxt, string) [pe_TimeStr](#) (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSUTC)
- #define **pe_TimeOfDayAndDiff**(pctxt, string) [pe_TimeStr](#) (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSDIFF)
- #define **pe_TimeOfDayAndFraction**(pctxt, string, n) [pe_TimeStr](#) (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|(n))
- #define **pe_TimeOfDayUtcAndFraction**(pctxt, string, n) [pe_TimeStr](#) (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSUTC|(n))
- #define **pe_TimeOfDayAndDiffAndFraction**(pctxt, string, n) [pe_TimeStr](#) (pctxt, string, OSHOURS|OSMINUTES|OSSECONDS|OSDIFF|(n))
- #define **pe_Year**(pctxt, string) [pe_DateStr](#) (pctxt, string, OSYEAR)
- #define **pe_YearDay**(pctxt, string) [pe_DateStr](#) (pctxt, string, OSYEAR|OSDAY)
- #define **pe_YearMonth**(pctxt, string) [pe_DateStr](#) (pctxt, string, OSYEAR|OSMONTH)
- #define **pe_YearMonthDay**(pctxt, string) [pe_DateStr](#) (pctxt, string, OSYEAR|OSMONTH|OSDAY)
- #define **pe_YearWeek**(pctxt, string) [pe_DateStr](#) (pctxt, string, OSYEAR|OSWEEK)
- #define **pe_YearWeekDay**(pctxt, string) [pe_DateStr](#) (pctxt, string, OSYEAR|OSWEEK|OSDAY)

Typedefs

- typedef struct [PERField](#) **PERField**

Functions

- int [pd_BigInteger](#) (OSCTXT *pctxt, const char **ppvalue)
- int [pd_BigIntegerEx](#) (OSCTXT *pctxt, const char **ppvalue, int radix)
- int [pd_BigIntegerValue](#) (OSCTXT *pctxt, const char **ppvalue, int radix, OSUINT32 nbytes)
- int [pd_BitString](#) (OSCTXT *pctxt, OSUINT32 *numbits_p, OSOCTET *buffer, OSSIZE bufsiz)
- int [pd_BitString64](#) (OSCTXT *pctxt, OSSIZE *numbits_p, OSOCTET *buffer, OSSIZE bufsiz)
- int [pd_BitString32](#) (OSCTXT *pctxt, ASN1BitStr32 *pbitstr, OSSIZE lower, OSSIZE upper)
- int [pd_BMPString](#) (OSCTXT *pctxt, ASN1BMPString *pvalue, Asn116BitCharSet *permCharSet)
- int [pd_UniversalString](#) (OSCTXT *pctxt, ASN1UniversalString *pvalue, Asn132BitCharSet *permCharSet)
- int [pd_byte_align](#) (OSCTXT *pctxt)
- int [pd_ChoiceOpenTypeExt](#) (OSCTXT *pctxt, const OSOCTET **object_p2, OSSIZE *pnumocts)
- int [pd_ConstInteger](#) (OSCTXT *pctxt, OSINT32 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstInt8](#) (OSCTXT *pctxt, OSINT8 *pvalue, OSINT64 lower, OSINT64 upper)

- int [pd_ConstInt16](#) (OSCTXT *pctx, OSINT16 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstInt64](#) (OSCTXT *pctx, OSINT64 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstUnsigned](#) (OSCTXT *pctx, OSUINT32 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd_ConstUnsignedSignedBound](#) (OSCTXT *pctx, OSUINT32 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstUInt8](#) (OSCTXT *pctx, OSUINT8 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd_ConstUInt8SignedBound](#) (OSCTXT *pctx, OSUINT8 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstUInt16](#) (OSCTXT *pctx, OSUINT16 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd_ConstUInt16SignedBound](#) (OSCTXT *pctx, OSUINT16 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstUInt64](#) (OSCTXT *pctx, OSUINT64 *pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd_ConstUInt64SignedBound](#) (OSCTXT *pctx, OSUINT64 *pvalue, OSINT64 lower, OSINT64 upper)
- int [pd_ConstWholeNumber](#) (OSCTXT *pctx, OSUINT32 *adjusted_value, OSUINT32 range_value)
- int [pd_ConstWholeNumber64](#) (OSCTXT *pctx, OSUINT64 *adjusted_value, OSUINT64 range_value)
- int [pd_ConstrainedString](#) (OSCTXT *pctx, const char **string, Asn1CharSet *pCharSet)
- int [pd_ConstrainedStringEx](#) (OSCTXT *pctx, const char **string, const char *charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int [pd_ConstrFixedLenStringEx](#) (OSCTXT *pctx, char *strbuf, size_t bufsiz, const char *charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int [pd_16BitConstrainedString](#) (OSCTXT *pctx, Asn116BitCharString *pString, Asn116BitCharSet *pCharSet)
- int [pd_32BitConstrainedString](#) (OSCTXT *pctx, Asn132BitCharString *pString, Asn132BitCharSet *pCharSet)
- int [pd_DateStr](#) (OSCTXT *pctx, const char **string, OSUINT32 flags)
- int [pd_DateTimeStr](#) (OSCTXT *pctx, const char **string, OSUINT32 flags)
- int [pd_Duration](#) (OSCTXT *pctx, const char **string, OSBOOL rec)
- int [pd_DynBitString](#) (OSCTXT *pctx, ASN1DynBitStr *pBitStr)
- int [pd_DynBitString64](#) (OSCTXT *pctx, ASN1DynBitStr64 *pBitStr)
- int [pd_DynOctetString](#) (OSCTXT *pctx, ASN1DynOctStr *pOctStr)
- int [pd_DynOctetString64](#) (OSCTXT *pctx, OSDynOctStr64 *pOctStr)
- int [pd_GetBinStrDataOffset](#) (OSCTXT *pctx, OSUINT32 *numbits, OSBOOL bitStrFlag)
- int [pd_GetComponentLength](#) (OSCTXT *pctx, OSUINT32 itemBits)
- int [pd_GetComponentLength64](#) (OSCTXT *pctx, OSUINT32 itemBits, OSSIZE *plength)
- int [pd_Interval](#) (OSCTXT *pctx, const char **string, OSBOOL rec, OSUINT32 startFlags, OSUINT32 endFlags)
- int [pd_Length](#) (OSCTXT *pctx, OSUINT32 *pvalue)
- int [pd_Length64](#) (OSCTXT *pctx, OSSIZE *pvalue)
- int [pd_ObjectIdentifier](#) (OSCTXT *pctx, ASN1OBJID *pvalue)
- int [pd_oid64](#) (OSCTXT *pctx, ASN1OID64 *pvalue)
- int [pd_RelativeOID](#) (OSCTXT *pctx, ASN1OBJID *pvalue)
- int [pd_OctetString](#) (OSCTXT *pctx, OSUINT32 *numocts, OSOCTET *buffer, OSUINT32 bufsiz)
- int [pd_OctetString64](#) (OSCTXT *pctx, OSSIZE *numocts, OSOCTET *buffer, OSSIZE bufsiz)
- int [pd_OpenType](#) (OSCTXT *pctx, const OSOCTET **object_p2, OSSIZE *numocts)
- int [pd_OpenTypeExt](#) (OSCTXT *pctx, const OSOCTET **object_p2, OSSIZE *numocts)
- int [pd_Real](#) (OSCTXT *pctx, OSREAL *pvalue)
- int [pd_SmallLength](#) (OSCTXT *pctx, OSUINT32 *pvalue)
- int [pd_SmallNonNegWholeNumber](#) (OSCTXT *pctx, OSUINT32 *pvalue)
- int [pd_SemiConsInteger](#) (OSCTXT *pctx, OSINT32 *pvalue, OSINT64 lower)
- int [pd_SemiConsUnsigned](#) (OSCTXT *pctx, OSUINT32 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUnsignedSignedBound](#) (OSCTXT *pctx, OSINT32 *pvalue, OSINT64 lower)
- int [pd_SemiConsInt8](#) (OSCTXT *pctx, OSINT8 *pvalue, OSINT64 lower)
- int [pd_SemiConsUInt8](#) (OSCTXT *pctx, OSUINT8 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUInt8SignedBound](#) (OSCTXT *pctx, OSUINT8 *pvalue, OSINT64 lower)
- int [pd_SemiConsInt16](#) (OSCTXT *pctx, OSINT16 *pvalue, OSINT64 lower)
- int [pd_SemiConsUInt16](#) (OSCTXT *pctx, OSUINT16 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUInt16SignedBound](#) (OSCTXT *pctx, OSUINT16 *pvalue, OSINT64 lower)

- int [pd_SemiConsInt64](#) (OSCTXT *pctxt, OSINT64 *pvalue, OSINT64 lower)
- int [pd_SemiConsUInt64](#) (OSCTXT *pctxt, OSUINT64 *pvalue, OSUINT64 lower)
- int [pd_SemiConsUInt64SignedBound](#) (OSCTXT *pctxt, OSUINT64 *pvalue, OSINT64 lower)
- int [pd_TimeStr](#) (OSCTXT *pctxt, const char **string, OSUINT32 flags)
- int [pd_UnconsInteger](#) (OSCTXT *pctxt, OSINT32 *pvalue)
- int [pd_UnconsLength](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- EXTPERMETHOD int [pd_UnconsLength64](#) (OSCTXT *pctxt, OSSIZE *pvalue)
- int [pd_UnconsUnsigned](#) (OSCTXT *pctxt, OSUINT32 *pvalue)
- int [pd_UnconsInt8](#) (OSCTXT *pctxt, OSINT8 *pvalue)
- int [pd_UnconsUInt8](#) (OSCTXT *pctxt, OSUINT8 *pvalue)
- int [pd_UnconsInt16](#) (OSCTXT *pctxt, OSINT16 *pvalue)
- int [pd_UnconsUInt16](#) (OSCTXT *pctxt, OSUINT16 *pvalue)
- int [pd_UnconsInt64](#) (OSCTXT *pctxt, OSINT64 *pvalue)
- int [pd_UnconsUInt64](#) (OSCTXT *pctxt, OSUINT64 *pvalue)
- int [pd_VarWidthCharString](#) (OSCTXT *pctxt, const char **pvalue)
- int [pd_YearInt](#) (OSCTXT *pctxt, OSINT32 *pvalue)
- int [pd_Real10](#) (OSCTXT *pctxt, const char **ppvalue)
- OSBOOL [pd_isFragmented](#) (OSCTXT *pctxt)
- void [pd_OpenTypeStart](#) (OSCTXT *pctxt, OSSIZE *pSavedSize, OSINT16 *pSavedBitOff)
- int [pd_OpenTypeEnd](#) (OSCTXT *pctxt, OSSIZE savedSize, OSINT16 savedBitOff)
- int [uperDecConstrString](#) (OSCTXT *pctxt, const char **string, const char *charSet, OSUINT32 nbits, OSUINT32 canSetBits)
- int [uperDecConstrFixedLenString](#) (OSCTXT *pctxt, char *strbuf, size_t bufsiz, const char *charSet, OSUINT32 nbits, OSUINT32 canSetBits)
- int [pe_16BitConstrainedString](#) (OSCTXT *pctxt, Asn116BitCharString value, Asn116BitCharSet *pCharSet)
- int [pe_32BitConstrainedString](#) (OSCTXT *pctxt, Asn132BitCharString value, Asn132BitCharSet *pCharSet)
- int [pe_2sCompBinInt](#) (OSCTXT *pctxt, OSINT32 value)
- int [pe_2sCompBinInt64](#) (OSCTXT *pctxt, OSINT64 value)
- int [pe_aligned_octets](#) (OSCTXT *pctxt, OSOCTET *pvalue, OSUINT32 nocts)
- int [pe_BigInteger](#) (OSCTXT *pctxt, const char *pvalue)
- int [pe_bits64](#) (OSCTXT *pctxt, OSUINT64 value, OSUINT32 nbits)
- int [pe_BitString](#) (OSCTXT *pctxt, OSSIZE numbits, const OSOCTET *data)
- int [pe_BitString32](#) (OSCTXT *pctxt, ASN1BitStr32 *pbitstr, OSUINT32 lower, OSUINT32 upper)
- EXTPERMETHOD int [pe_BitStringExt](#) (OSCTXT *pctxt, OSSIZE numbits, const OSOCTET *data, OSSIZE dataSize, const OSOCTET *extData)
- int [pe_BMPString](#) (OSCTXT *pctxt, ASN1BMPString value, Asn116BitCharSet *permCharSet)
- int [pe_UniversalString](#) (OSCTXT *pctxt, ASN1UniversalString value, Asn132BitCharSet *permCharSet)
- int [pe_byte_align](#) (OSCTXT *pctxt)
- int [pe_ChoiceTypeExt](#) (OSCTXT *pctxt, OSUINT32 numocts, const OSOCTET *data)
- int [pe_ConstInt64](#) (OSCTXT *pctxt, OSINT64 value, OSINT64 lower, OSINT64 upper)
- int [pe_ConstrainedString](#) (OSCTXT *pctxt, const char *string, Asn1CharSet *pCharSet)
- int [pe_ConstrainedStringEx](#) (OSCTXT *pctxt, const char *string, const char *charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int [pe_ConsUInt64](#) (OSCTXT *pctxt, OSUINT64 value, OSUINT64 lower, OSUINT64 upper)
- int [pe_ConsUInt64SignedBound](#) (OSCTXT *pctxt, OSUINT64 value, OSINT64 lower, OSINT64 upper)
- int [pe_ConsWholeNumber](#) (OSCTXT *pctxt, OSUINT32 adjusted_value, OSUINT32 range_value)
- int [pe_ConsWholeNumber64](#) (OSCTXT *pctxt, OSUINT64 adjusted_value, OSUINT64 range_value)
- int [pe_DateStr](#) (OSCTXT *pctxt, const char *string, OSUINT32 flags)
- int [pe_DateTimeStr](#) (OSCTXT *pctxt, const char *string, OSUINT32 flags)
- int [pe_Duration](#) (OSCTXT *pctxt, const char *string, OSBOOL rec)
- OSUINT32 [pe_GetIntLen](#) (OSUINT32 value)

- `size_t pe_GetMsgBitCnt` (OSCTXT *pctx)
- OSOCTET * `pe_GetMsgPtr` (OSCTXT *pctx, OSINT32 *pLength)
- OSOCTET * `pe_GetMsgPtrU` (OSCTXT *pctx, OSUINT32 *pLength)
- OSOCTET * `pe_GetMsgPtr64` (OSCTXT *pctx, OSSIZE *pLength)
- int `pe_Interval` (OSCTXT *pctx, const char *string, OSBOOL rec, OSUINT32 startFlags, OSUINT32 endFlags)
- int `pe_Length` (OSCTXT *pctx, OSSIZE value)
- int `pe_NonNegBinInt` (OSCTXT *pctx, OSUINT32 value)
- int `pe_NonNegBinInt64` (OSCTXT *pctx, OSUINT64 value)
- int `pe_ObjectIdentifier` (OSCTXT *pctx, ASN1OBJID *pvalue)
- int `pe_oid64` (OSCTXT *pctx, ASN1OID64 *pvalue)
- int `pe_RelativeOID` (OSCTXT *pctx, ASN1OBJID *pvalue)
- int `pe_OctetString` (OSCTXT *pctx, OSSIZE numocts, const OSOCTET *data)
- int `pe_OpenType` (OSCTXT *pctx, OSSIZE numocts, const OSOCTET *data)
- int `pe_OpenTypeEnd` (OSCTXT *pctx, OSUINT32 pos, void *pPerField)
- int `pe_OpenTypeExt` (OSCTXT *pctx, OSRTDList *pElemList)
- int `pe_OpenTypeExtBits` (OSCTXT *pctx, OSRTDList *pElemList)
- int `pe_OpenTypeStart` (OSCTXT *pctx, OSUINT32 *pPos, void **ppPerField)
- int `pe_Real` (OSCTXT *pctx, OSREAL value)
- int `pe_SmallNonNegWholeNumber` (OSCTXT *pctx, OSUINT32 value)
- int `pe_SmallLength` (OSCTXT *pctx, OSSIZE value)
- int `pe_SemiConsInt64` (OSCTXT *pctx, OSINT64 value, OSINT64 lower)
- int `pe_SemiConsUnsignedSignedBound` (OSCTXT *pctx, OSUINT32 value, OSINT64 lower)
- int `pe_SemiConsUInt64` (OSCTXT *pctx, OSUINT64 value, OSUINT64 lower)
- int `pe_SemiConsUInt64SignedBound` (OSCTXT *pctx, OSUINT64 value, OSINT64 lower)
- int `pe_TimeStr` (OSCTXT *pctx, const char *string, OSUINT32 flags)
- int `pe_UnconsLength` (OSCTXT *pctx, OSSIZE value)
- int `pe_UnconsInteger` (OSCTXT *pctx, OSINT32 value)
- int `pe_UnconsInt64` (OSCTXT *pctx, OSINT64 value)
- int `pe_UnconsUnsigned` (OSCTXT *pctx, OSUINT32 value)
- int `pe_UnconsUInt64` (OSCTXT *pctx, OSUINT64 value)
- int `pe_VarWidthCharString` (OSCTXT *pctx, const char *value)
- int `pe_YearInt` (OSCTXT *pctx, OSINT32 value)
- int `pe_Real10` (OSCTXT *pctx, const char *pvalue)
- int `uperEncConstrString` (OSCTXT *pctx, const char *string, const char *charSet, OSUINT32 nbits, OSUINT32 canSetBits)
- int `pu_addSizeConstraint` (OSCTXT *pctx, const Asn1SizeCnst *pSize)
- OSBOOL `pu_alignCharStr` (OSCTXT *pctx, OSUINT32 len, OSUINT32 nbits, Asn1SizeCnst *pSize)
- OSUINT32 `pu_bitcnt` (OSUINT32 value)
- Asn1SizeCnst * `pu_checkSize` (Asn1SizeCnst *pSizeList, OSUINT32 value, OSBOOL *pExtendable)
- int `pu_checkSizeExt` (Asn1SizeCnst *pSizeCnst, OSSIZE value, OSBOOL *pExtendable, Asn1SizeValueRange *pSizeRange, OSBOOL *pExtSize)
- void `pu_freeContext` (OSCTXT *pctx)
- `size_t pu_getMaskAndIndex` (`size_t` bitOffset, unsigned char *pMask)
- `size_t pu_getMsgLen` (OSCTXT *pctx)
- `size_t pu_getMsgLenBits` (OSCTXT *pctx)
- void `pu_hexdump` (OSCTXT *pctx)
- int `pu_setBuffer` (OSCTXT *pctx, OSOCTET *bufaddr, `size_t` bufsiz, OSBOOL aligned)
- int `pe_resetBuffer` (OSCTXT *pctx)
- int `pu_initContext` (OSCTXT *pctx, OSOCTET *bufaddr, OSUINT32 bufsiz, OSBOOL aligned)
- int `pu_initContextBuffer` (OSCTXT *pTarget, OSCTXT *pSource)
- const char * `pu_getFullName` (OSCTXT *pctx, const char *suffix)

- void `pu_insLenField` (OSCTXT *pctx)
- OSBOOL `pu_isFixedSize` (const Asn1SizeCnst *pSizeList)
- OSCTXT * `pu_newContext` (OSOCTET *bufaddr, OSUINT32 bufsiz, OSBOOL aligned)
- `PERField` * `pu_newField` (OSCTXT *pctx, const char *nameSuffix)
- void `pu_initFieldList` (OSCTXT *pctx, OSINT16 bitOffset)
- void `pu_initRtxDiagBitFieldList` (OSCTXT *pctx, OSINT16 bitOffset)
- void `pu_popName` (OSCTXT *pctx)
- void `pu_pushElemName` (OSCTXT *pctx, int idx)
- void `pu_pushName` (OSCTXT *pctx, const char *name)
- void `pu_setCharSet` (Asn1CharSet *pCharSet, const char *permSet)
- int `pu_set16BitCharSet` (OSCTXT *pctx, Asn116BitCharSet *pCharSet, Asn116BitCharSet *pAlphabet)
- void `pu_set16BitCharSetFromRange` (Asn116BitCharSet *pCharSet, OSUINT16 firstChar, OSUINT16 lastChar)
- void `pu_setFldBitCount` (OSCTXT *pctx)
- void `pu_setFldBitOffset` (OSCTXT *pctx)
- void `pu_setFldListFromCtx` (OSCTXT *pctx, OSCTXT *srcctx)
- void `pu_setOpenTypeFldList` (OSCTXT *pctx, OSRTSList *plist)
- void `pu_setRtxDiagOpenTypeFldList` (OSRTDiagBitFieldList *pMainBFList, OSRTDiagBitFieldList *pOpenTypeBFList)
- OSBOOL `pu_setTrace` (OSCTXT *pctx, OSBOOL value)
- void `pu_setAligned` (OSCTXT *pctx, OSBOOL value)
- void `pu_deleteFieldList` (OSCTXT *pctx)
- OSBOOL `pu_BitAndOctetStringAlignmentTest` (const Asn1SizeCnst *pSizeCnst, OSSIZE itemCount, OSBOL bitStrFlag)
- void `pu_bindump` (OSCTXT *pctx, const char *varname)
- void `pu_dumpField` (OSCTXT *pctx, `PERField` *pField, const char *varname, size_t nextBitOffset, `BinDumpBuffer` *pbuf)
- int `pu_set32BitCharSet` (OSCTXT *pctx, Asn132BitCharSet *pCharSet, Asn132BitCharSet *pAlphabet)
- void `pu_set32BitCharSetFromRange` (Asn132BitCharSet *pCharSet, OSUINT32 firstChar, OSUINT32 lastChar)
- int `pu_GetLibVersion` (OSVOIDARG)
- const char * `pu_GetLibInfo` (OSVOIDARG)
- int `pu_checkSizeConstraint` (OSCTXT *pctx, int size)
- Asn1SizeCnst * `pu_getSizeConstraint` (OSCTXT *pctx, OSBOOL extbit)
- int `pu_getBitOffset` (OSCTXT *pctx)
- void `pu_setBitOffset` (OSCTXT *pctx, int bitOffset)

8.1.1 Detailed Description

ASN.1 runtime constants, data structure definitions, and functions to support the Packed Encoding Rules (PER) as defined in the ITU-T X.691 standard.

8.2 `asn1PerCppType.h` File Reference

```
#include "rtpersrc/asn1per.h"
#include "rtsrc/asn1CppType.h"
#include "rtxsrc/rtxBitEncode.h"
```

Classes

- class [ASN1PERMessageBuffer](#)
- class [ASN1PEREncodeBuffer](#)
- class [ASN1PERDecodeBuffer](#)

8.2.1 Detailed Description

PER C++ type and class definitions.

Index

- ASN1PERDecodeBuffer, 119
 - ASN1PERDecodeBuffer, 120, 121
 - byteAlign, 122
 - isA, 122
 - peekByte, 122
 - readBinaryFile, 123
 - readBytes, 123
- ASN1PEREncodeBuffer, 124
 - ASN1PEREncodeBuffer, 125, 126
 - byteAlign, 126
 - encodeBit, 126
 - encodeBits, 127
 - getMsgBitCnt, 127
 - getMsgCopy, 127
 - getMsgPtr, 128
 - init, 128
 - isA, 128
- ASN1PERMessageBuffer, 129
 - ASN1PERMessageBuffer, 130, 131
 - binDump, 131
 - getMsgLen, 132
 - hexDump, 132
 - isAligned, 132
 - setBuffer, 133
 - setTrace, 133
- asn1PerCppType.h, 144
- asn1per.h, 135
- binDump
 - ASN1PERMessageBuffer, 131
- BinDumpBuffer, 134
- byteAlign
 - ASN1PERDecodeBuffer, 122
 - ASN1PEREncodeBuffer, 126
- encodeBit
 - ASN1PEREncodeBuffer, 126
- encodeBits
 - ASN1PEREncodeBuffer, 127
- getMsgBitCnt
 - ASN1PEREncodeBuffer, 127
- getMsgCopy
 - ASN1PEREncodeBuffer, 127
- getMsgLen
 - ASN1PERMessageBuffer, 132
- getMsgPtr
 - ASN1PEREncodeBuffer, 128
- hexDump
 - ASN1PERMessageBuffer, 132
- init
 - ASN1PEREncodeBuffer, 128
- isAligned
 - ASN1PERMessageBuffer, 132
- isA
 - ASN1PERDecodeBuffer, 122
 - ASN1PEREncodeBuffer, 128
- PD_BYTE_ALIGN0
 - PER Runtime Library Functions., 18
- PD_CHECKSEQOFLLEN
 - PER Runtime Library Functions., 18
- PER C Decode Functions., 22
 - pd_16BitConstrainedString, 24
 - pd_32BitConstrainedString, 25
 - pd_BMPString, 28
 - pd_BigInteger, 25
 - pd_BigIntegerEx, 26
 - pd_BigIntegerValue, 27
 - pd_BitString, 27
 - pd_BitString32, 28
 - pd_BitString64, 28
 - pd_ChoiceOpenTypeExt, 29
 - pd_ConsInt16, 30
 - pd_ConsInt64, 30
 - pd_ConsInt8, 31
 - pd_ConsInteger, 31
 - pd_ConsUInt16, 34
 - pd_ConsUInt16SignedBound, 34
 - pd_ConsUInt64, 35
 - pd_ConsUInt64SignedBound, 36
 - pd_ConsUInt8, 36
 - pd_ConsUInt8SignedBound, 37
 - pd_ConsUnsigned, 37
 - pd_ConsUnsignedSignedBound, 38
 - pd_ConsWholeNumber, 38
 - pd_ConsWholeNumber64, 39
 - pd_ConstrFixedLenStringEx, 33
 - pd_ConstrainedString, 32
 - pd_ConstrainedStringEx, 33

pd_DateStr, 39
 pd_DateTimeStr, 40
 pd_Duration, 40
 pd_DynBitString, 41
 pd_DynBitString64, 41
 pd_DynOctetString, 41
 pd_DynOctetString64, 42
 pd_GetBinStrDataOffset, 42
 pd_GetComponentLength, 43
 pd_GetComponentLength64, 43
 pd_Interval, 44
 pd_Length, 45
 pd_Length64, 45
 pd_ObjectIdentifier, 46
 pd_OctetString, 46
 pd_OctetString64, 47
 pd_OpenType, 48
 pd_OpenTypeExt, 48
 pd_Real, 49
 pd_Real10, 50
 pd_RelativeOID, 50
 pd_SemiConsInt16, 51
 pd_SemiConsInt64, 51
 pd_SemiConsInt8, 52
 pd_SemiConsInteger, 52
 pd_SemiConsUInt16, 53
 pd_SemiConsUInt16SignedBound, 53
 pd_SemiConsUInt64, 54
 pd_SemiConsUInt64SignedBound, 54
 pd_SemiConsUInt8, 55
 pd_SemiConsUInt8SignedBound, 55
 pd_SemiConsUnsigned, 56
 pd_SemiConsUnsignedSignedBound, 56
 pd_SmallLength, 57
 pd_SmallNonNegWholeNumber, 57
 pd_TimeStr, 58
 pd_UnconsInt16, 58
 pd_UnconsInt64, 59
 pd_UnconsInt8, 59
 pd_UnconsInteger, 60
 pd_UnconsLength, 60
 pd_UnconsLength64, 61
 pd_UnconsUInt16, 61
 pd_UnconsUInt64, 61
 pd_UnconsUInt8, 62
 pd_UnconsUnsigned, 62
 pd_UniversalString, 63
 pd_VarWidthCharString, 64
 pd_YearInt, 64
 pd_byte_align, 29
 pd_isFragmented, 44
 pd_moveBitCursor, 24
 pd_oid64, 47
 uperDecConstrFixedLenString, 65
 uperDecConstrString, 65
 PER C Encode Functions., 67
 pe_16BitConstrainedString, 73
 pe_2sCompBinInt, 74
 pe_2sCompBinInt64, 74
 pe_32BitConstrainedString, 75
 pe_BMPString, 79
 pe_BigInteger, 77
 pe_BitString, 78
 pe_BitStringExt, 78
 pe_CheckBuffer, 69
 pe_ChoiceTypeExt, 80
 pe_ConsInt64, 80
 pe_ConsInteger, 70
 pe_ConsUInt64, 82
 pe_ConsUInt64SignedBound, 83
 pe_ConsUnsigned, 70
 pe_ConsUnsignedSignedBound, 71
 pe_ConsWholeNumber, 83
 pe_ConsWholeNumber64, 84
 pe_ConstrainedString, 81
 pe_ConstrainedStringEx, 82
 pe_DateStr, 85
 pe_DateTimeStr, 85
 pe_Duration, 86
 pe_GetIntLen, 86
 pe_GetMsgBitCnt, 86
 pe_GetMsgPtr, 87
 pe_GetMsgPtr64, 87
 pe_GetMsgPtrU, 87
 pe_Interval, 88
 pe_Length, 88
 pe_NonNegBinInt, 89
 pe_NonNegBinInt64, 89
 pe_ObjectIdentifier, 90
 pe_OctetString, 90
 pe_OpenType, 91
 pe_OpenTypeEnd, 92
 pe_OpenTypeExt, 92
 pe_OpenTypeExtBits, 93
 pe_OpenTypeStart, 93
 pe_Real, 95
 pe_Real10, 95
 pe_RelativeOID, 96
 pe_SemiConsInt64, 96
 pe_SemiConsInteger, 72
 pe_SemiConsUInt64, 97
 pe_SemiConsUInt64SignedBound, 97
 pe_SemiConsUnsigned, 73
 pe_SemiConsUnsignedSignedBound, 98
 pe_SmallLength, 98
 pe_SmallNonNegWholeNumber, 99
 pe_TimeStr, 99
 pe_UnconsInt64, 100

- pe_UnconsInteger, [100](#)
- pe_UnconsLength, [101](#)
- pe_UnconsUInt64, [101](#)
- pe_UnconsUnsigned, [102](#)
- pe_UniversalString, [102](#)
- pe_VarWidthCharString, [103](#)
- pe_YearInt, [103](#)
- pe_aligned_octets, [75](#)
- pe_bit, [68](#)
- pe_bits, [69](#)
- pe_bits64, [77](#)
- pe_byte_align, [79](#)
- pe_octets, [72](#)
- pe_oid64, [91](#)
- uperEncConstrString, [104](#)
- PER C Utility Functions, [105](#)
 - pe_resetBuffer, [106](#)
 - pu_GetLibInfo, [108](#)
 - pu_GetLibVersion, [108](#)
 - pu_addSizeConstraint, [106](#)
 - pu_bindump, [107](#)
 - pu_checkSizeExt, [107](#)
 - pu_freeContext, [108](#)
 - pu_getMsgLen, [108](#)
 - pu_getMsgLenBits, [109](#)
 - pu_hexdump, [109](#)
 - pu_initContext, [110](#)
 - pu_initContextBuffer, [110](#)
 - pu_initFieldList, [111](#)
 - pu_initRtxDiagBitFieldList, [111](#)
 - pu_insLenField, [112](#)
 - pu_isFixedSize, [112](#)
 - pu_newContext, [112](#)
 - pu_newField, [113](#)
 - pu_popName, [113](#)
 - pu_pushElemName, [114](#)
 - pu_pushName, [114](#)
 - pu_set16BitCharSet, [114](#)
 - pu_set16BitCharSetFromRange, [115](#)
 - pu_set32BitCharSet, [115](#)
 - pu_set32BitCharSetFromRange, [116](#)
 - pu_setBuffer, [116](#)
 - pu_setCharSet, [117](#)
- PER C++ Runtime Classes., [11](#)
- PER Message Buffer Classes, [12](#)
- PER Runtime Library Functions., [13](#)
 - PD_BYTE_ALIGN0, [18](#)
 - PD_CHECKSEQOFLen, [18](#)
 - PU_GETPADBITS, [20](#)
 - PU_GETSIZECONSTRAINT, [20](#)
 - PU_SETCHARSET, [20](#)
 - PU_SETCTXTBITOFFSET, [21](#)
 - PU_SETSIZECONSTRAINT, [21](#)
 - pd_DateTime, [19](#)
 - pd_NumericString, [19](#)
 - pd_bit, [18](#)
 - pe_DateTime, [19](#)
 - pe_NumericString, [20](#)
 - PERField, [134](#)
 - PU_GETPADBITS
 - PER Runtime Library Functions., [20](#)
 - PU_GETSIZECONSTRAINT
 - PER Runtime Library Functions., [20](#)
 - PU_SETCHARSET
 - PER Runtime Library Functions., [20](#)
 - PU_SETCTXTBITOFFSET
 - PER Runtime Library Functions., [21](#)
 - PU_SETSIZECONSTRAINT
 - PER Runtime Library Functions., [21](#)
 - pd_16BitConstrainedString
 - PER C Decode Functions., [24](#)
 - pd_32BitConstrainedString
 - PER C Decode Functions., [25](#)
 - pd_BMPString
 - PER C Decode Functions., [28](#)
 - pd_BigInteger
 - PER C Decode Functions., [25](#)
 - pd_BigIntegerEx
 - PER C Decode Functions., [26](#)
 - pd_BigIntegerValue
 - PER C Decode Functions., [27](#)
 - pd_BitString
 - PER C Decode Functions., [27](#)
 - pd_BitString32
 - PER C Decode Functions., [28](#)
 - pd_BitString64
 - PER C Decode Functions., [28](#)
 - pd_ChoiceOpenTypeExt
 - PER C Decode Functions., [29](#)
 - pd_ConstInt16
 - PER C Decode Functions., [30](#)
 - pd_ConstInt64
 - PER C Decode Functions., [30](#)
 - pd_ConstInt8
 - PER C Decode Functions., [31](#)
 - pd_ConstInteger
 - PER C Decode Functions., [31](#)
 - pd_ConstUInt16
 - PER C Decode Functions., [34](#)
 - pd_ConstUInt16SignedBound
 - PER C Decode Functions., [34](#)
 - pd_ConstUInt64
 - PER C Decode Functions., [35](#)
 - pd_ConstUInt64SignedBound
 - PER C Decode Functions., [36](#)
 - pd_ConstUInt8
 - PER C Decode Functions., [36](#)
 - pd_ConstUInt8SignedBound

PER C Decode Functions., 37
 pd_ConsUnsigned
 PER C Decode Functions., 37
 pd_ConsUnsignedSignedBound
 PER C Decode Functions., 38
 pd_ConsWholeNumber
 PER C Decode Functions., 38
 pd_ConsWholeNumber64
 PER C Decode Functions., 39
 pd_ConstrFixedLenStringEx
 PER C Decode Functions., 33
 pd_ConstrainedString
 PER C Decode Functions., 32
 pd_ConstrainedStringEx
 PER C Decode Functions., 33
 pd_DateStr
 PER C Decode Functions., 39
 pd_DateTime
 PER Runtime Library Functions., 19
 pd_DateTimeStr
 PER C Decode Functions., 40
 pd_Duration
 PER C Decode Functions., 40
 pd_DynBitString
 PER C Decode Functions., 41
 pd_DynBitString64
 PER C Decode Functions., 41
 pd_DynOctetString
 PER C Decode Functions., 41
 pd_DynOctetString64
 PER C Decode Functions., 42
 pd_GetBinStrDataOffset
 PER C Decode Functions., 42
 pd_GetComponentLength
 PER C Decode Functions., 43
 pd_GetComponentLength64
 PER C Decode Functions., 43
 pd_Interval
 PER C Decode Functions., 44
 pd_Length
 PER C Decode Functions., 45
 pd_Length64
 PER C Decode Functions., 45
 pd_NumericString
 PER Runtime Library Functions., 19
 pd_ObjectIdentifier
 PER C Decode Functions., 46
 pd_OctetString
 PER C Decode Functions., 46
 pd_OctetString64
 PER C Decode Functions., 47
 pd_OpenType
 PER C Decode Functions., 48
 pd_OpenTypeExt
 PER C Decode Functions., 48
 pd_Real
 PER C Decode Functions., 49
 pd_Real10
 PER C Decode Functions., 50
 pd_RelativeOID
 PER C Decode Functions., 50
 pd_SemiConsInt16
 PER C Decode Functions., 51
 pd_SemiConsInt64
 PER C Decode Functions., 51
 pd_SemiConsInt8
 PER C Decode Functions., 52
 pd_SemiConsInteger
 PER C Decode Functions., 52
 pd_SemiConsUInt16
 PER C Decode Functions., 53
 pd_SemiConsUInt16SignedBound
 PER C Decode Functions., 53
 pd_SemiConsUInt64
 PER C Decode Functions., 54
 pd_SemiConsUInt64SignedBound
 PER C Decode Functions., 54
 pd_SemiConsUInt8
 PER C Decode Functions., 55
 pd_SemiConsUInt8SignedBound
 PER C Decode Functions., 55
 pd_SemiConsUnsigned
 PER C Decode Functions., 56
 pd_SemiConsUnsignedSignedBound
 PER C Decode Functions., 56
 pd_SmallLength
 PER C Decode Functions., 57
 pd_SmallNonNegWholeNumber
 PER C Decode Functions., 57
 pd_TimeStr
 PER C Decode Functions., 58
 pd_UnconsInt16
 PER C Decode Functions., 58
 pd_UnconsInt64
 PER C Decode Functions., 59
 pd_UnconsInt8
 PER C Decode Functions., 59
 pd_UnconsInteger
 PER C Decode Functions., 60
 pd_UnconsLength
 PER C Decode Functions., 60
 pd_UnconsLength64
 PER C Decode Functions., 61
 pd_UnconsUInt16
 PER C Decode Functions., 61
 pd_UnconsUInt64
 PER C Decode Functions., 61
 pd_UnconsUInt8

PER C Decode Functions., 62
 pd_UnconsUnsigned
 PER C Decode Functions., 62
 pd_UniversalString
 PER C Decode Functions., 63
 pd_VarWidthCharString
 PER C Decode Functions., 64
 pd_YearInt
 PER C Decode Functions., 64
 pd_bit
 PER Runtime Library Functions., 18
 pd_byte_align
 PER C Decode Functions., 29
 pd_isFragmented
 PER C Decode Functions., 44
 pd_moveBitCursor
 PER C Decode Functions., 24
 pd_oid64
 PER C Decode Functions., 47
 pe_16BitConstrainedString
 PER C Encode Functions., 73
 pe_2sCompBinInt
 PER C Encode Functions., 74
 pe_2sCompBinInt64
 PER C Encode Functions., 74
 pe_32BitConstrainedString
 PER C Encode Functions., 75
 pe_BMPString
 PER C Encode Functions., 79
 pe_BigInteger
 PER C Encode Functions., 77
 pe_BitString
 PER C Encode Functions., 78
 pe_BitStringExt
 PER C Encode Functions., 78
 pe_CheckBuffer
 PER C Encode Functions., 69
 pe_ChoiceTypeExt
 PER C Encode Functions., 80
 pe_ConstInt64
 PER C Encode Functions., 80
 pe_ConstInteger
 PER C Encode Functions., 70
 pe_ConstUInt64
 PER C Encode Functions., 82
 pe_ConstUInt64SignedBound
 PER C Encode Functions., 83
 pe_ConstUnsigned
 PER C Encode Functions., 70
 pe_ConstUnsignedSignedBound
 PER C Encode Functions., 71
 pe_ConstWholeNumber
 PER C Encode Functions., 83
 pe_ConstWholeNumber64
 PER C Encode Functions., 84
 pe_ConstrainedString
 PER C Encode Functions., 81
 pe_ConstrainedStringEx
 PER C Encode Functions., 82
 pe_DateStr
 PER C Encode Functions., 85
 pe_DateTime
 PER Runtime Library Functions., 19
 pe_DateTimeStr
 PER C Encode Functions., 85
 pe_Duration
 PER C Encode Functions., 86
 pe_GetIntLen
 PER C Encode Functions., 86
 pe_GetMsgBitCnt
 PER C Encode Functions., 86
 pe_GetMsgPtr
 PER C Encode Functions., 87
 pe_GetMsgPtr64
 PER C Encode Functions., 87
 pe_GetMsgPtrU
 PER C Encode Functions., 87
 pe_Interval
 PER C Encode Functions., 88
 pe_Length
 PER C Encode Functions., 88
 pe_NonNegBinInt
 PER C Encode Functions., 89
 pe_NonNegBinInt64
 PER C Encode Functions., 89
 pe_NumericString
 PER Runtime Library Functions., 20
 pe_ObjectIdentifier
 PER C Encode Functions., 90
 pe_OctetString
 PER C Encode Functions., 90
 pe_OpenType
 PER C Encode Functions., 91
 pe_OpenTypeEnd
 PER C Encode Functions., 92
 pe_OpenTypeExt
 PER C Encode Functions., 92
 pe_OpenTypeExtBits
 PER C Encode Functions., 93
 pe_OpenTypeStart
 PER C Encode Functions., 93
 pe_Real
 PER C Encode Functions., 95
 pe_Real10
 PER C Encode Functions., 95
 pe_RelativeOID
 PER C Encode Functions., 96
 pe_SemiConstInt64

PER C Encode Functions., 96
 pe_SemiConsInteger
 PER C Encode Functions., 72
 pe_SemiConsUInt64
 PER C Encode Functions., 97
 pe_SemiConsUInt64SignedBound
 PER C Encode Functions., 97
 pe_SemiConsUnsigned
 PER C Encode Functions., 73
 pe_SemiConsUnsignedSignedBound
 PER C Encode Functions., 98
 pe_SmallLength
 PER C Encode Functions., 98
 pe_SmallNonNegWholeNumber
 PER C Encode Functions., 99
 pe_TimeStr
 PER C Encode Functions., 99
 pe_UnconsInt64
 PER C Encode Functions., 100
 pe_UnconsInteger
 PER C Encode Functions., 100
 pe_UnconsLength
 PER C Encode Functions., 101
 pe_UnconsUInt64
 PER C Encode Functions., 101
 pe_UnconsUnsigned
 PER C Encode Functions., 102
 pe_UniversalString
 PER C Encode Functions., 102
 pe_VarWidthCharString
 PER C Encode Functions., 103
 pe_YearInt
 PER C Encode Functions., 103
 pe_aligned_octets
 PER C Encode Functions., 75
 pe_bit
 PER C Encode Functions., 68
 pe_bits
 PER C Encode Functions., 69
 pe_bits64
 PER C Encode Functions., 77
 pe_byte_align
 PER C Encode Functions., 79
 pe_octets
 PER C Encode Functions., 72
 pe_oid64
 PER C Encode Functions., 91
 pe_resetBuffer
 PER C Utility Functions, 106
 peekByte
 ASN1PERDecodeBuffer, 122
 pu_GetLibInfo
 PER C Utility Functions, 108
 pu_GetLibVersion
 PER C Utility Functions, 108
 pu_addSizeConstraint
 PER C Utility Functions, 106
 pu_bindump
 PER C Utility Functions, 107
 pu_checkSizeExt
 PER C Utility Functions, 107
 pu_freeContext
 PER C Utility Functions, 108
 pu_getMsgLen
 PER C Utility Functions, 108
 pu_getMsgLenBits
 PER C Utility Functions, 109
 pu_hexdump
 PER C Utility Functions, 109
 pu_initContext
 PER C Utility Functions, 110
 pu_initContextBuffer
 PER C Utility Functions, 110
 pu_initFieldList
 PER C Utility Functions, 111
 pu_initRtxDiagBitFieldList
 PER C Utility Functions, 111
 pu_insLenField
 PER C Utility Functions, 112
 pu_isFixedSize
 PER C Utility Functions, 112
 pu_newContext
 PER C Utility Functions, 112
 pu_newField
 PER C Utility Functions, 113
 pu_popName
 PER C Utility Functions, 113
 pu_pushElemName
 PER C Utility Functions, 114
 pu_pushName
 PER C Utility Functions, 114
 pu_set16BitCharSet
 PER C Utility Functions, 114
 pu_set16BitCharSetFromRange
 PER C Utility Functions, 115
 pu_set32BitCharSet
 PER C Utility Functions, 115
 pu_set32BitCharSetFromRange
 PER C Utility Functions, 116
 pu_setBuffer
 PER C Utility Functions, 116
 pu_setCharSet
 PER C Utility Functions, 117
 readBinaryFile
 ASN1PERDecodeBuffer, 123
 readBytes
 ASN1PERDecodeBuffer, 123

setBuffer
 ASN1PERMessageBuffer, [133](#)
setTrace
 ASN1PERMessageBuffer, [133](#)

uperDecConstrFixedLenString
 PER C Decode Functions., [65](#)
uperDecConstrString
 PER C Decode Functions., [65](#)
uperEncConstrString
 PER C Encode Functions., [104](#)