

# ASN1C

---

ASN.1 Compiler  
Version 7.1  
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 ©1997–2017 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](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>ASNIC PER Runtime Classes and Library Functions</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>2</b>
2.1	Modules . . . . .	2
<b>3</b>	<b>Class Index</b>	<b>3</b>
3.1	Class Hierarchy . . . . .	3
<b>4</b>	<b>Class Index</b>	<b>4</b>
4.1	Class List . . . . .	4
<b>5</b>	<b>File Index</b>	<b>5</b>
5.1	File List . . . . .	5
<b>6</b>	<b>Module Documentation</b>	<b>6</b>
6.1	PER C++ Runtime Classes. . . . .	6
6.2	PER Message Buffer Classes . . . . .	7
6.2.1	Detailed Description . . . . .	7
6.3	PER Runtime Library Functions. . . . .	8
6.3.1	Detailed Description . . . . .	13
6.3.2	Define Documentation . . . . .	13
6.3.2.1	pd_bit . . . . .	13
6.3.2.2	PD_BYTE_ALIGN0 . . . . .	13
6.3.2.3	PD_CHECKSEQOFLN . . . . .	13
6.3.2.4	pd_DateTime . . . . .	14
6.3.2.5	pd_NumericString . . . . .	14
6.3.2.6	pe_DateTime . . . . .	14
6.3.2.7	pe_NumericString . . . . .	14
6.3.2.8	PU_GETPADBITS . . . . .	14
6.3.2.9	PU_GETSIZECONSTRAINT . . . . .	14

6.3.2.10	PU_SETCHARSET	14
6.3.2.11	PU_SETCTXTBITOFFSET	15
6.3.2.12	PU_SETSIZECONSTRAINT	15
6.4	PER C Decode Functions.	16
6.4.1	Detailed Description	18
6.4.2	Define Documentation	18
6.4.2.1	pd_moveBitCursor	18
6.4.3	Function Documentation	18
6.4.3.1	pd_16BitConstrainedString	18
6.4.3.2	pd_32BitConstrainedString	18
6.4.3.3	pd_BigInteger	19
6.4.3.4	pd_BigIntegerEx	19
6.4.3.5	pd_BigIntegerValue	20
6.4.3.6	pd_BitString	20
6.4.3.7	pd_BitString32	20
6.4.3.8	pd_BitString64	21
6.4.3.9	pd_BMPString	21
6.4.3.10	pd_byte_align	21
6.4.3.11	pd_ChoiceOpenTypeExt	22
6.4.3.12	pd_ConsInt16	22
6.4.3.13	pd_ConsInt64	22
6.4.3.14	pd_ConsInt8	23
6.4.3.15	pd_ConsInteger	23
6.4.3.16	pd_ConstrainedString	23
6.4.3.17	pd_ConstrainedStringEx	24
6.4.3.18	pd_ConstrFixedLenStringEx	24
6.4.3.19	pd_ConsUInt16	25
6.4.3.20	pd_ConsUInt16SignedBound	25
6.4.3.21	pd_ConsUInt64	25
6.4.3.22	pd_ConsUInt64SignedBound	26
6.4.3.23	pd_ConsUInt8	26
6.4.3.24	pd_ConsUInt8SignedBound	26
6.4.3.25	pd_ConsUnsigned	27
6.4.3.26	pd_ConsUnsignedSignedBound	27
6.4.3.27	pd_ConsWholeNumber	27
6.4.3.28	pd_ConsWholeNumber64	28
6.4.3.29	pd_DateStr	28

6.4.3.30	pd_DateTimeStr	28
6.4.3.31	pd_Duration	29
6.4.3.32	pd_DynBitString	29
6.4.3.33	pd_DynBitString64	29
6.4.3.34	pd_DynOctetString	30
6.4.3.35	pd_DynOctetString64	30
6.4.3.36	pd_GetBinStrDataOffset	30
6.4.3.37	pd_GetComponentLength	30
6.4.3.38	pd_GetComponentLength64	31
6.4.3.39	pd_Interval	31
6.4.3.40	pd_isFragmented	32
6.4.3.41	pd_Length	32
6.4.3.42	pd_Length64	32
6.4.3.43	pd_ObjectIdentifier	32
6.4.3.44	pd_OctetString	33
6.4.3.45	pd_OctetString64	33
6.4.3.46	pd_oid64	34
6.4.3.47	pd_OpenType	34
6.4.3.48	pd_OpenTypeExt	34
6.4.3.49	pd_Real	35
6.4.3.50	pd_Real10	35
6.4.3.51	pd_RelativeOID	35
6.4.3.52	pd_SemiConsInt16	36
6.4.3.53	pd_SemiConsInt64	36
6.4.3.54	pd_SemiConsInt8	36
6.4.3.55	pd_SemiConsInteger	37
6.4.3.56	pd_SemiConsUInt16	37
6.4.3.57	pd_SemiConsUInt16SignedBound	37
6.4.3.58	pd_SemiConsUInt64	38
6.4.3.59	pd_SemiConsUInt64SignedBound	38
6.4.3.60	pd_SemiConsUInt8	38
6.4.3.61	pd_SemiConsUInt8SignedBound	39
6.4.3.62	pd_SemiConsUnsigned	39
6.4.3.63	pd_SemiConsUnsignedSignedBound	39
6.4.3.64	pd_SmallLength	40
6.4.3.65	pd_SmallNonNegWholeNumber	40
6.4.3.66	pd_TimeStr	40

6.4.3.67	pd_UnconsInt16	41
6.4.3.68	pd_UnconsInt64	41
6.4.3.69	pd_UnconsInt8	41
6.4.3.70	pd_UnconsInteger	42
6.4.3.71	pd_UnconsLength	42
6.4.3.72	pd_UnconsLength64	42
6.4.3.73	pd_UnconsUInt16	43
6.4.3.74	pd_UnconsUInt64	43
6.4.3.75	pd_UnconsUInt8	43
6.4.3.76	pd_UnconsUnsigned	44
6.4.3.77	pd_UniversalString	44
6.4.3.78	pd_VarWidthCharString	44
6.4.3.79	pd_YearInt	45
6.4.3.80	uperDecConstrFixedLenString	45
6.4.3.81	uperDecConstrString	45
6.5	PER C Encode Functions	47
6.5.1	Detailed Description	48
6.5.2	Define Documentation	48
6.5.2.1	pe_bit	48
6.5.2.2	pe_bits	49
6.5.2.3	pe_CheckBuffer	49
6.5.2.4	pe_ConsInteger	49
6.5.2.5	pe_ConsUnsigned	50
6.5.2.6	pe_ConsUnsignedSignedBound	50
6.5.2.7	pe_octets	51
6.5.2.8	pe_SemiConsInteger	51
6.5.2.9	pe_SemiConsUnsigned	51
6.5.3	Function Documentation	52
6.5.3.1	pe_16BitConstrainedString	52
6.5.3.2	pe_2sCompBinInt	52
6.5.3.3	pe_2sCompBinInt64	52
6.5.3.4	pe_32BitConstrainedString	53
6.5.3.5	pe_aligned_octets	53
6.5.3.6	pe_BigInteger	53
6.5.3.7	pe_bits64	54
6.5.3.8	pe_BitString	54
6.5.3.9	pe_BMPString	54



6.5.3.10	pe_byte_align	55
6.5.3.11	pe_ChoiceTypeExt	55
6.5.3.12	pe_ConsInt64	55
6.5.3.13	pe_ConstrainedString	56
6.5.3.14	pe_ConstrainedStringEx	56
6.5.3.15	pe_ConsUInt64	56
6.5.3.16	pe_ConsUInt64SignedBound	57
6.5.3.17	pe_ConsWholeNumber	57
6.5.3.18	pe_ConsWholeNumber64	58
6.5.3.19	pe_DateStr	58
6.5.3.20	pe_DateTimeStr	58
6.5.3.21	pe_Duration	59
6.5.3.22	pe_GetIntLen	59
6.5.3.23	pe_GetMsgBitCnt	59
6.5.3.24	pe_GetMsgPtr	59
6.5.3.25	pe_GetMsgPtr64	60
6.5.3.26	pe_GetMsgPtrU	60
6.5.3.27	pe_Interval	60
6.5.3.28	pe_Length	60
6.5.3.29	pe_NonNegBinInt	61
6.5.3.30	pe_NonNegBinInt64	61
6.5.3.31	pe_ObjectIdentifier	61
6.5.3.32	pe_OctetString	62
6.5.3.33	pe_oid64	62
6.5.3.34	pe_OpenType	62
6.5.3.35	pe_OpenTypeEnd	63
6.5.3.36	pe_OpenTypeExt	63
6.5.3.37	pe_OpenTypeExtBits	63
6.5.3.38	pe_OpenTypeStart	64
6.5.3.39	pe_Real	64
6.5.3.40	pe_Real10	64
6.5.3.41	pe_RelativeOID	65
6.5.3.42	pe_SemiConsInt64	65
6.5.3.43	pe_SemiConsUInt64	65
6.5.3.44	pe_SemiConsUInt64SignedBound	66
6.5.3.45	pe_SemiConsUnsignedSignedBound	66
6.5.3.46	pe_SmallLength	66

6.5.3.47	<code>pe_SmallNonNegWholeNumber</code>	67
6.5.3.48	<code>pe_TimeStr</code>	67
6.5.3.49	<code>pe_UnconsInt64</code>	67
6.5.3.50	<code>pe_UnconsInteger</code>	68
6.5.3.51	<code>pe_UnconsLength</code>	68
6.5.3.52	<code>pe_UnconsUInt64</code>	68
6.5.3.53	<code>pe_UnconsUnsigned</code>	68
6.5.3.54	<code>pe_UniversalString</code>	69
6.5.3.55	<code>pe_VarWidthCharString</code>	69
6.5.3.56	<code>pe_YearInt</code>	70
6.5.3.57	<code>uperEncConstrString</code>	70
6.6	PER C Utility Functions	71
6.6.1	Detailed Description	72
6.6.2	Function Documentation	72
6.6.2.1	<code>pu_addSizeConstraint</code>	72
6.6.2.2	<code>pu_bindump</code>	72
6.6.2.3	<code>pu_checkSizeExt</code>	72
6.6.2.4	<code>pu_freeContext</code>	73
6.6.2.5	<code>pu_GetLibInfo</code>	73
6.6.2.6	<code>pu_GetLibVersion</code>	73
6.6.2.7	<code>pu_getMsgLen</code>	73
6.6.2.8	<code>pu_getMsgLenBits</code>	74
6.6.2.9	<code>pu_hexdump</code>	74
6.6.2.10	<code>pu_initContext</code>	74
6.6.2.11	<code>pu_initContextBuffer</code>	75
6.6.2.12	<code>pu_initFieldList</code>	75
6.6.2.13	<code>pu_initRtxDiagBitFieldList</code>	75
6.6.2.14	<code>pu_insLenField</code>	76
6.6.2.15	<code>pu_isFixedSize</code>	76
6.6.2.16	<code>pu_newContext</code>	76
6.6.2.17	<code>pu_newField</code>	76
6.6.2.18	<code>pu_popName</code>	77
6.6.2.19	<code>pu_pushElemName</code>	77
6.6.2.20	<code>pu_pushName</code>	77
6.6.2.21	<code>pu_set16BitCharSet</code>	77
6.6.2.22	<code>pu_set16BitCharSetFromRange</code>	78
6.6.2.23	<code>pu_set32BitCharSet</code>	78

6.6.2.24	pu_set32BitCharSetFromRange	78
6.6.2.25	pu_setBuffer	78
6.6.2.26	pu_setCharSet	79
<b>7</b>	<b>Class Documentation</b>	<b>80</b>
7.1	ASN1PERDecodeBuffer Class Reference	80
7.1.1	Detailed Description	80
7.1.2	Constructor & Destructor Documentation	81
7.1.2.1	ASN1PERDecodeBuffer	81
7.1.2.2	ASN1PERDecodeBuffer	81
7.1.2.3	ASN1PERDecodeBuffer	81
7.1.2.4	ASN1PERDecodeBuffer	81
7.1.2.5	ASN1PERDecodeBuffer	82
7.1.3	Member Function Documentation	82
7.1.3.1	byteAlign	82
7.1.3.2	isA	82
7.1.3.3	peekByte	82
7.1.3.4	readBinaryFile	83
7.1.3.5	readBytes	83
7.2	ASN1PEREncodeBuffer Class Reference	84
7.2.1	Detailed Description	84
7.2.2	Constructor & Destructor Documentation	84
7.2.2.1	ASN1PEREncodeBuffer	84
7.2.2.2	ASN1PEREncodeBuffer	85
7.2.2.3	ASN1PEREncodeBuffer	85
7.2.2.4	ASN1PEREncodeBuffer	85
7.2.3	Member Function Documentation	85
7.2.3.1	byteAlign	85
7.2.3.2	encodeBit	86
7.2.3.3	encodeBits	86
7.2.3.4	getMsgBitCnt	86
7.2.3.5	getMsgCopy	86
7.2.3.6	getMsgPtr	87
7.2.3.7	init	87
7.2.3.8	isA	87
7.3	ASN1PERMessageBuffer Class Reference	88
7.3.1	Detailed Description	88

7.3.2	Constructor & Destructor Documentation	88
7.3.2.1	ASN1PERMessageBuffer	88
7.3.2.2	ASN1PERMessageBuffer	89
7.3.2.3	ASN1PERMessageBuffer	89
7.3.2.4	ASN1PERMessageBuffer	89
7.3.3	Member Function Documentation	90
7.3.3.1	binDump	90
7.3.3.2	getMsgLen	90
7.3.3.3	hexDump	90
7.3.3.4	isAligned	90
7.3.3.5	setBuffer	90
7.3.3.6	setTrace	91
7.4	BinDumpBuffer Struct Reference	92
7.5	PERField Struct Reference	93
<b>8</b>	<b>File Documentation</b>	<b>94</b>
8.1	asn1per.h File Reference	94
8.1.1	Detailed Description	103
8.2	asn1PerCppType.h File Reference	104
8.2.1	Detailed Description	104

# 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. . . . .	6
PER Message Buffer Classes . . . . .	7
PER Runtime Library Functions. . . . .	8
PER C Decode Functions. . . . .	16
PER C Encode Functions. . . . .	47
PER C Utility Functions . . . . .	71

# Chapter 3

## Class Index

### 3.1 Class Hierarchy

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

ASN1PERMessageBuffer . . . . .	88
ASN1PERDecodeBuffer . . . . .	80
ASN1PEREncodeBuffer . . . . .	84
BinDumpBuffer . . . . .	92
PERField . . . . .	93

# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">ASN1PERDecodeBuffer</a> . . . . .	80
<a href="#">ASN1PEREncodeBuffer</a> . . . . .	84
<a href="#">ASN1PERMessageBuffer</a> . . . . .	88
<a href="#">BinDumpBuffer</a> . . . . .	92
<a href="#">PERField</a> . . . . .	93



# Chapter 5

## File Index

### 5.1 File List

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

<a href="#">asn1per.h</a> .....	94
<a href="#">asn1PerCppType.h</a> .....	104

# Chapter 6

## Module Documentation

### 6.1 PER C++ Runtime Classes.

#### Modules

- [PER Message Buffer Classes](#)

## 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 ASN.1 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.

### Classes

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

### Modules

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

### Defines

- #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**(pMainBFLList, pOpenTypeBFLList)
- #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)
- #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, 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\_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|OSUTC|(n))
- #define **pe\_MinutesAndDiffAndFraction**(pctxt, string, n) pe\_TimeStr (pctxt, string, OSHOURS|OSMINUTES|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, OSDURATION)
- #define **pe\_RecStartTimeDurationInterval**(pctxt, string, flags) pe\_Interval (pctxt, string, TRUE, flags, OSDURATION)
- #define **pe\_RecStartDateTimeDurationInterval**(pctxt, string, flags) pe\_Interval (pctxt, string, TRUE, flags, OSDURATION)
- #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 **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 Define Documentation

#### 6.3.2.1 #define pd\_bit(pctxt, pvalue) rtxDecBit(pctxt,pvalue)

perutil

#### 6.3.2.2 #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 #define PD\_CHECKSEQOFLEN(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 #define pd\_DateTime(pctxt, string)

##### Value:

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

#### 6.3.2.5 #define pd\_NumericString(pctxt, pvalue, permCharSet)

##### Value:

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

#### 6.3.2.6 #define pe\_DateTime(pctxt, string)

##### Value:

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

#### 6.3.2.7 #define pe\_NumericString(pctxt, value, permCharSet)

##### Value:

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

#### 6.3.2.8 #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 #define PU\_GETSIZECONSTRAINT(pctxt, extbit)

##### Value:

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

#### 6.3.2.10 #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 #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 #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.

### Defines

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

### 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_ConsInteger` (OSCTXT \*pctxt, OSINT32 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsInt8` (OSCTXT \*pctxt, OSINT8 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsInt16` (OSCTXT \*pctxt, OSINT16 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsInt64` (OSCTXT \*pctxt, OSINT64 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsUnsigned` (OSCTXT \*pctxt, OSUINT32 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConsUnsignedSignedBound` (OSCTXT \*pctxt, OSUINT32 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsUInt8` (OSCTXT \*pctxt, OSUINT8 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConsUInt8SignedBound` (OSCTXT \*pctxt, OSUINT8 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsUInt16` (OSCTXT \*pctxt, OSUINT16 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConsUInt16SignedBound` (OSCTXT \*pctxt, OSUINT16 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsUInt64` (OSCTXT \*pctxt, OSUINT64 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int `pd_ConsUInt64SignedBound` (OSCTXT \*pctxt, OSUINT64 \*pvalue, OSINT64 lower, OSINT64 upper)
- int `pd_ConsWholeNumber` (OSCTXT \*pctxt, OSUINT32 \*padjusted\_value, OSUINT32 range\_value)
- int `pd_ConsWholeNumber64` (OSCTXT \*pctxt, OSUINT64 \*padjusted\_value, OSUINT64 range\_value)
- int `pd_ConstrainedString` (OSCTXT \*pctxt, const char \*\*string, Asn1CharSet \*pCharSet)
- int `pd_ConstrainedStringEx` (OSCTXT \*pctxt, const char \*\*string, const char \*charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int `pd_ConstrFixedLenStringEx` (OSCTXT \*pctxt, char \*strbuf, size\_t bufsiz, const char \*charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int `pd_16BitConstrainedString` (OSCTXT \*pctxt, Asn116BitCharString \*pString, Asn116BitCharSet \*pCharSet)
- int `pd_32BitConstrainedString` (OSCTXT \*pctxt, Asn132BitCharString \*pString, Asn132BitCharSet \*pCharSet)
- int `pd_DateStr` (OSCTXT \*pctxt, const char \*\*string, OSUINT32 flags)
- int `pd_DateTimeStr` (OSCTXT \*pctxt, const char \*\*string, OSUINT32 flags)
- int `pd_Duration` (OSCTXT \*pctxt, const char \*\*string, OSBOOL rec)
- int `pd_DynBitString` (OSCTXT \*pctxt, ASN1DynBitStr \*pBitStr)
- int `pd_DynBitString64` (OSCTXT \*pctxt, ASN1DynBitStr64 \*pBitStr)
- int `pd_DynOctetString` (OSCTXT \*pctxt, ASN1DynOctStr \*pOctStr)
- int `pd_DynOctetString64` (OSCTXT \*pctxt, OSDynOctStr64 \*pOctStr)
- int `pd_GetBinStrDataOffset` (OSCTXT \*pctxt, OSUINT32 \*pnumbits, OSBOOL bitStrFlag)
- int `pd_GetComponentLength` (OSCTXT \*pctxt, 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 \*pnumocts, OSOCTET \*buffer, OSUINT32 bufsiz)
- int [pd\\_OctetString64](#) (OSCTXT \*pctx, OSSIZE \*pnumocts, OSOCTET \*buffer, OSSIZE bufsiz)
- int [pd\\_OpenType](#) (OSCTXT \*pctx, const OSOCTET \*\*object\_p2, OSSIZE \*pnumocts)
- int [pd\\_OpenTypeExt](#) (OSCTXT \*pctx, const OSOCTET \*\*object\_p2, OSSIZE \*pnumocts)
- 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 \*pctx, OSINT64 \*pvalue, OSINT64 lower)
- int [pd\\_SemiConsUInt64](#) (OSCTXT \*pctx, OSUINT64 \*pvalue, OSUINT64 lower)
- int [pd\\_SemiConsUInt64SignedBound](#) (OSCTXT \*pctx, OSUINT64 \*pvalue, OSINT64 lower)
- int [pd\\_TimeStr](#) (OSCTXT \*pctx, const char \*\*string, OSUINT32 flags)
- int [pd\\_UnconsInteger](#) (OSCTXT \*pctx, OSINT32 \*pvalue)
- int [pd\\_UnconsLength](#) (OSCTXT \*pctx, OSUINT32 \*pvalue)
- EXTPERMETHOD int [pd\\_UnconsLength64](#) (OSCTXT \*pctx, OSSIZE \*pvalue)
- int [pd\\_UnconsUnsigned](#) (OSCTXT \*pctx, OSUINT32 \*pvalue)
- int [pd\\_UnconsInt8](#) (OSCTXT \*pctx, OSINT8 \*pvalue)
- int [pd\\_UnconsUInt8](#) (OSCTXT \*pctx, OSUINT8 \*pvalue)
- int [pd\\_UnconsInt16](#) (OSCTXT \*pctx, OSINT16 \*pvalue)
- int [pd\\_UnconsUInt16](#) (OSCTXT \*pctx, OSUINT16 \*pvalue)
- int [pd\\_UnconsInt64](#) (OSCTXT \*pctx, OSINT64 \*pvalue)
- int [pd\\_UnconsUInt64](#) (OSCTXT \*pctx, OSUINT64 \*pvalue)
- int [pd\\_VarWidthCharString](#) (OSCTXT \*pctx, const char \*\*pvalue)
- int [pd\\_YearInt](#) (OSCTXT \*pctx, OSINT32 \*pvalue)
- int [pd\\_Real10](#) (OSCTXT \*pctx, const char \*\*pvalue)
- OSBOOL [pd\\_isFragmented](#) (OSCTXT \*pctx)
- void [pd\\_OpenTypeStart](#) (OSCTXT \*pctx, OSSIZE \*pSavedSize, OSINT16 \*pSavedBitOff)
- int [pd\\_OpenTypeEnd](#) (OSCTXT \*pctx, OSSIZE savedSize, OSINT16 savedBitOff)
- int [uperDecConstrString](#) (OSCTXT \*pctx, const char \*\*string, const char \*charSet, OSUINT32 nbits, OSUINT32 canSetBits)
- int [uperDecConstrFixedLenString](#) (OSCTXT \*pctx, 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 Define Documentation

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

#### Parameters

*pctxt* 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.

*bitOffset* The bit offset inside the message buffer.

## 6.4.3 Function Documentation

### 6.4.3.1 `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

*pctxt* 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.

*pString* 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.

*pCharSet* 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 `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

*pctxt* 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.

*pString* 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.

*pCharSet* 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 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 `rtxBigIntegerSetStr` / `rtxBigIntegerToString` functions.

## Parameters

*pctxt* Pointer to context block structure.

*ppvalue* 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 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

*pctxt* Pointer to context block structure.

*ppvalue* 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.

*radix* 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 `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

*pctxt* Pointer to context block structure.

*ppvalue* 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.

*radix* Radix to be used for decoded string. Valid values are 2, 8, 10, or 16.

*nbytes* Length in bytes of the value component.

##### Returns

Completion status of operation:

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

#### 6.4.3.6 `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

*pctxt* 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.

*numbits\_p* Pointer to an unsigned integer variable to receive decoded number of bits.

*buffer* Pointer to a fixed-size or pre-allocated array of `bufsiz` octets to receive a decoded bit string.

*bufsiz* 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 `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 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 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

*pctxt* 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.

*pvalue* 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.

*permCharSet* 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 int pd\_byte\_align (OSCTXT \* *pctxt*)

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

##### Parameters

*pctxt* 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 ASN1PERDecodeBuffer::byteAlign().

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

##### Parameters

*pctxt* 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.

*object\_p2* A pointer to an open type variable to receive the decoded data.

*pnumocts* A pointer to an unsigned buffer of `bufsiz` octets to receive decoded data.

#### 6.4.3.12 `int pd_ConsInt16 (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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 16-bit integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### Returns

Completion status of operation:

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

#### 6.4.3.13 `int pd_ConsInt64 (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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 64-bit integer variable to receive decoded value.

*lower* Lower range value, represented as 64-bit integer.

*upper* 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 `int pd_ConstInt8 (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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 8-bit integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### Returns

Completion status of operation:

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

#### 6.4.3.15 `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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### Returns

Completion status of operation:

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

#### 6.4.3.16 `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

*pctxt* Pointer to context block structure.

*string* Pointer to const char\* to receive decoded string. Memory will be allocated for this variable using internal memory management functions.

*pCharSet* 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 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**

*pctxt* Pointer to context block structure.

*string* Pointer to const char\* to receive decoded string. Memory will be allocated for this variable using internal memory management functions.

*charSet* String containing permitted alphabet character set. Can be null if no character set was specified.

*abits* Number of bits in a character set character (aligned).

*ubits* Number of bits in a character set character (unaligned).

*canSetBits* 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 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**

*pctxt* Pointer to context block structure.

*strbuf* Pointer to character array to receive decoded string.

*bufsiz* Size of *strbuf* character array.

*charSet* String containing permitted alphabet character set. Can be null if no character set was specified.

*abits* Number of bits in a character set character (aligned).

*ubits* Number of bits in a character set character (unaligned).

*canSetBits* 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 `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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 16-bit unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### Returns

Completion status of operation:

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

#### 6.4.3.20 `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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 16-bit unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### Returns

Completion status of operation:

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

#### 6.4.3.21 `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

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 64-bit unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### Returns

Completion status of operation:

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

#### 6.4.3.22 **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**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 64-bit unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### **Returns**

Completion status of operation:

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

#### 6.4.3.23 **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**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 8-bit unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### **Returns**

Completion status of operation:

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

#### 6.4.3.24 **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**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 8-bit unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### **Returns**

Completion status of operation:

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

#### 6.4.3.25 **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**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### **Returns**

Completion status of operation:

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

#### 6.4.3.26 **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**

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned integer variable to receive decoded value.

*lower* Lower range value.

*upper* Upper range value.

##### **Returns**

Completion status of operation:

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

#### 6.4.3.27 **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**

*pctxt* Pointer to context block structure.

*padjusted\_value* 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.

*range\_value* 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 `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

*pctxt* Pointer to context block structure.

*padjusted\_value* 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.

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

This function will decode an ISO 8601 DATE type.

##### Parameters

*pctxt* Pointer to context block structure.

*string* Pointer to string variable to receive decoded value in string form (YYYY:MM:DD) and ect.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*string* Pointer to string variable to receive decoded value in string form (YYYY-MM-DDTHH:MM:SS) and ect.

*flags* 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 int pd\_Duration (OSCTXT \* *pctxt*, const char \*\* *string*, OSBOOL *rec*)

This function will decode an ISO 8601 DURATION types.

##### Parameters

*pctxt* Pointer to context block structure.

*string* Pointer to string variable to receive decoded value in string form (PnYnMnDTnHnMnS).

*rec* Decode recursive interval (Rn/).

##### Returns

Completion status of operation:

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

#### 6.4.3.32 int pd\_DynBitString (OSCTXT \* *pctxt*, 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

*pctxt* 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.

*pBitStr* 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 rtxMemAlloc function. This memory is tracked within the context and released when the pu\_freeContext function is invoked.

##### Returns

Completion status of operation:

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

#### 6.4.3.33 int pd\_DynBitString64 (OSCTXT \* *pctxt*, ASN1DynBitStr64 \* *pBitStr*)

64-bit version of pd\_DynBitString.

##### See also

[pd\\_DynBitString](#)

#### 6.4.3.34 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

*pctxt* 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.

*pOctStr* 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 int pd\_DynOctetString64 (OSCTXT \* *pctxt*, OSDynOctStr64 \* *pOctStr*)

64-bit version of pd\_DynOctetString.

##### See also

[pd\\_DynOctetString](#)

#### 6.4.3.36 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

*pctxt* 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.

*pnumbits* A pointer to an unsigned integer to receive the bit count value.

*bitStrFlag* 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 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

*pctxt* 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.

*itemBits* The size of the specific entity.

## Returns

Completion status of operation:

- $\geq 0$  = success, total parsed length
- negative return value is error.

### 6.4.3.38 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

*pctxt* 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.

*itemBits* The size of the specific entity.

*plength* 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 int pd\_Interval (OSCTXT \* *pctxt*, const char \*\* *string*, OSBOOL *rec*, OSUINT32 *startFlags*, OSUINT32 *endFlags*)

This function will decode an ISO 8601 INTERVAL type.

## Parameters

*pctxt* Pointer to context block structure.

*string* Pointer to string variable to receive decoded value in string form (start/end).

*rec* Decode recursive interval (Rn/).

*startFlags* 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.

*endFlags* Set format flags of interval end.

## Returns

Completion status of operation:

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

#### 6.4.3.40 OSBOOL pd\_isFragmented (OSCTXT \* *pctxt*)

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

##### Parameters

*pctxt* Pointer to a context structure.

#### 6.4.3.41 int pd\_Length (OSCTXT \* *pctxt*, OSUINT32 \* *pvalue*)

This function will decode a length determinant value.

##### Parameters

*pctxt* 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.

*pvalue* 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 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

*pctxt* 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.

*pvalue* 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 int pd\_ObjectIdentifier (OSCTXT \* *pctxt*, ASN1OBJID \* *pvalue*)

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* 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.

*pnumocts* A pointer to an unsigned buffer of *bufsiz* octets to receive decoded data.

*buffer* A pointer to a pre-allocated buffer of size octets to receive the decoded data.

*bufsiz* 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 `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

*pctxt* 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.

*pnumocts* A pointer to an unsigned buffer of *bufsiz* octets to receive decoded data.

*buffer* A pointer to a pre-allocated buffer of size octets to receive the decoded data.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* 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.

*object\_p2* A pointer to an open type variable to receive the decoded data.

*pnumocts* 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 `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

*pctxt* 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.

*object\_p2* A pointer to an open type variable to receive the decoded data.

*pnumocts* 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 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

*pctxt* 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.

*pvalue* Pointer to an real variable to receive decoded value.

## Returns

Completion status of operation:

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

### 6.4.3.50 int pd\_Real10 (OSCTXT \* *pctxt*, const char \*\* *ppvalue*)

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

## Parameters

*pctxt* 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.

*ppvalue* 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 int pd\_RelativeOID (OSCTXT \* *pctxt*, ASN1OBJID \* *pvalue*)

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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 int pd\_SemiConsInt16 (OSCTXT \* *pctxt*, OSINT16 \* *pvalue*, OSINT64 *lower*)

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 64-bit integer variable to receive decoded value.

*lower* 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 int pd\_SemiConsInt64 (OSCTXT \* *pctxt*, OSINT64 \* *pvalue*, OSINT64 *lower*)

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 64-bit integer variable to receive decoded value.

*lower* 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 int pd\_SemiConsInt8 (OSCTXT \* *pctxt*, OSINT8 \* *pvalue*, OSINT64 *lower*)

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

## Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to 8-bit integer variable to receive decoded value.

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

This function will decode a semi-constrained integer.

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to integer variable to receive decoded value.

*lower* Lower range value, represented as signed integer.

##### Returns

Completion status of operation:

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 64-bit integer variable to receive decoded value.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 64-bit integer variable to receive decoded value.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 64-bit integer variable to receive decoded value.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 64-bit integer variable to receive decoded value.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 8-bit integer variable to receive decoded value.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned 8-bit integer variable to receive decoded value.

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

This function will decode a semi-constrained unsigned integer.

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned integer variable to receive decoded value.

*lower* Lower range value, represented as unsigned integer.

##### Returns

Completion status of operation:

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

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

This function will decode a semi-constrained unsigned integer.

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* Pointer to unsigned integer variable to receive decoded value.

*lower* Lower range value, represented as unsigned integer.

##### Returns

Completion status of operation:

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

#### 6.4.3.64 **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**

*pctxt* 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.

*pvalue* 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 **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**

*pctxt* 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.

*pvalue* 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 **int pd\_TimeStr (OSCTXT \* *pctxt*, const char \*\* *string*, OSUINT32 *flags*)**

This function will decode ISO 8601 TIME types.

##### **Parameters**

*pctxt* Pointer to context block structure.

*string* Pointer to string variable to receive decoded value in string form (HH:MM:SS) and ect.

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

This function will decode an unconstrained 16-bit integer.

##### Parameters

*pctxt* Pointer to context block structure.

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

This function will decode an unconstrained 64-bit integer.

##### Parameters

*pctxt* Pointer to context block structure.

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

This function will decode an unconstrained 8-bit integer.

##### Parameters

*pctxt* Pointer to context block structure.

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

This function will decode an unconstrained integer.

##### Parameters

*pctxt* Pointer to context block structure.

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* Pointer to context block structure.

*pvalue* 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 **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**

*pctxt* Pointer to context block structure.

*pvalue* 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 **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**

*pctxt* 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.

*pvalue* 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.

*permCharSet* 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 **int pd\_VarWidthCharString (OSCTXT \* *pctxt*, const char \*\* *pvalue*)**

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

##### **Parameters**

*pctxt* Pointer to context block structure.

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

This function will decode an ISO 8601 YEAR type.

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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 `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

*pctxt* Pointer to context block structure.

*strbuf* Pointer to character array to receive decoded string.

*bufsiz* Size of *strbuf* character array.

*charSet* String containing permitted alphabet character set. Can be null if no character set was specified.

*nbits* Number of bits in a character set character (unaligned).

*canSetBits* 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 `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

*pctxt* Pointer to context block structure.

*string* Pointer to const char\* to receive decoded string. Memory will be allocated for this variable using internal memory management functions.

*charSet* String containing permitted alphabet character set. Can be null if no character set was specified.

*nbits* Number of bits in a character set character (unaligned).

*canSetBits* 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.

### Defines

- #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)

### Functions

- 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 numocts, const OSOCTET \*data)
- int `pe_BitString32` (OSCTXT \*pctxt, ASN1BitStr32 \*pbitstr, OSUINT32 lower, OSUINT32 upper)
- 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_ConsInt64` (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 \*pctxt)
- OSOCTET \* `pe_GetMsgPtr` (OSCTXT \*pctxt, OSINT32 \*pLength)
- OSOCTET \* `pe_GetMsgPtrU` (OSCTXT \*pctxt, OSUINT32 \*pLength)
- OSOCTET \* `pe_GetMsgPtr64` (OSCTXT \*pctxt, OSSIZE \*pLength)
- int `pe_Interval` (OSCTXT \*pctxt, const char \*string, OSBOOL rec, OSUINT32 startFlags, OSUINT32 endFlags)
- int `pe_Length` (OSCTXT \*pctxt, OSSIZE value)
- int `pe_NonNegBinInt` (OSCTXT \*pctxt, OSUINT32 value)

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

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

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

## Parameters

*pctxt* 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.

*value* The BOOLEAN value to be encoded.

### 6.5.2.2 #define pe\_bits(pctxt, value, nbits) rtxEncBits(pctxt,value,nbits)

This function encodes multiple bits.

## Parameters

*pctxt* Pointer to context block structure.

*value* Unsigned integer containing the bits to be encoded.

*nbits* Number of bits in value to encode.

## Returns

Completion status of operation:

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

### 6.5.2.3 #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

*pctxt* 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.

*nbytes* 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 #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

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*lower* Lower range value.

*upper* Upper range value.

## Returns

Completion status of operation:

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

### 6.5.2.5 #define pe\_ConsUnsigned(pctxt, value, lower, upper) pe\_ConsUInt64(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  $\geq 0$ , and 2. The upper value of the range is  $\geq \text{MAXINT}$

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*lower* Lower range value.

*upper* Upper range value.

## Returns

Completion status of operation:

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

### 6.5.2.6 #define pe\_ConsUnsignedSignedBound(pctxt, value, lower, upper) pe\_ConsUInt64SignedBound(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  $\geq 0$ , and 2. The upper value of the range is  $\geq \text{MAXINT}$

## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*lower* Lower range value.

*upper* Upper range value.

## Returns

Completion status of operation:

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

#### 6.5.2.7 #define pe\_octets(pctx, pvalue, nbits) rtxEncBitsFromArray(pctx,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

*pctx* 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.

*pvalue* A pointer to an array of octets to encode

*nbits* The number of Octets to encode

##### Returns

Completion status of operation:

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

#### 6.5.2.8 #define pe\_SemiConsInteger(pctx, value, lower) pe\_SemiConsInt64(pctx, value, lower)

This function encodes an semi-constrained integer.

##### Parameters

*pctx* Pointer to context block structure.

*value* Value to be encoded.

*lower* Lower range value, represented as signed integer.

##### Returns

Completion status of operation:

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

#### 6.5.2.9 #define pe\_SemiConsUnsigned(pctx, value, lower) pe\_SemiConsUInt64(pctx, value, lower)

This function encodes an semi-constrained unsigned integer.

##### Parameters

*pctx* Pointer to context block structure.

*value* Value to be encoded.

*lower* 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 `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

*pctxt* 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.

*value* 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.

*pCharSet* 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 `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

*pctxt* Pointer to context block structure.

*value* Signed integer value to be encoded.

#### Returns

Completion status of operation:

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

### 6.5.3.3 `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

*pctxt* Pointer to context block structure.

*value* 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 `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

*pctxt* 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.

*value* 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.

*pCharSet* 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 `int pe_aligned_octets (OSCTXT * pctxt, OSOCTET * pvalue, OSUINT32 nocts)`

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* A pointer to a character string containing the value to be encoded.

*nocts* The number of octets.

#### 6.5.3.6 `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

*pctxt* Pointer to context block structure.

*pvalue* 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 int pe\_bits64 (OSCTXT \* *pctxt*, OSUINT64 *value*, OSUINT32 *nbits*)

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

#### Parameters

- pctxt* Pointer to context block structure.
- value* Unsigned 64-bit integer containing the bits to be encoded.
- nbits* Number of bits in value to encode.

#### Returns

Completion status of operation:

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

### 6.5.3.8 int pe\_BitString (OSCTXT \* *pctxt*, OSSIZE *numocts*, const OSOCTET \* *data*)

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

#### Parameters

- pctxt* 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.
- numocts* The number of bits n the string to be encoded.
- data* 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 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

- pctxt* 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.
- value* 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.
- permCharSet* 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.

### 6.5.3.10 int pe\_byte\_align (OSCTXT \* *pctxt*)

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

#### Parameters

*pctxt* 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().

### 6.5.3.11 int pe\_ChoiceTypeExt (OSCTXT \* *pctxt*, OSUINT32 *numocts*, const OSOCTET \* *data*)

#### Parameters

*pctxt* 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.

*numocts* Number of octets in the string to be encoded.

*data* Pointer to octet string data to be encoded.

#### Returns

Completion status of operation:

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

### 6.5.3.12 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

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as 64-bit integer.

*lower* Lower range value, represented as 64-bit integer.

*upper* Upper range value, represented as 64-bit integer.

#### Returns

Completion status of operation:

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

### 6.5.3.13 `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

- pctxt* Pointer to context block structure.
- string* Pointer to string to be encoded.
- pCharSet* Pointer to a character set descriptor structure.

#### Returns

Completion status of operation:

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

### 6.5.3.14 `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

- pctxt* Pointer to context block structure.
- string* Pointer to string to be encoded.
- charSet* String containing permitted alphabet character set. Can be null if no character set was specified.
- abits* Number of bits in a character set character (aligned).
- ubits* Number of bits in a character set character (unaligned).
- canSetBits* 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.

### 6.5.3.15 `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  $\geq 0$ , and 2. The upper value of the range is  $\geq \text{MAXINT}$

#### Parameters

- pctxt* Pointer to context block structure.
- value* Value to be encoded, represented as unsigned 64-bit integer.

*lower* Lower range value, represented as unsigned 64-bit integer.

*upper* Upper range value, represented as unsigned 64-bit integer.

### Returns

Completion status of operation:

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

#### 6.5.3.16 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  $\geq 0$ , and 2. The upper value of the range is  $\leq \text{MAXINT}$

### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as unsigned 64-bit integer.

*lower* Lower range value, represented as unsigned 64-bit integer.

*upper* 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 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

*pctxt* Pointer to context block structure.

*adjusted\_value* 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.

*range\_value* 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.

### 6.5.3.18 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

*pctxt* Pointer to context block structure.

*adjusted\_value* 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.

*range\_value* 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 int pe\_DateStr (OSCTXT \* *pctxt*, const char \* *string*, OSUINT32 *flags*)

This function will encode an ISO 8601 DATE types.

#### Parameters

*pctxt* Pointer to context block structure.

*string* Character string variable containing value to be encoded in string form (YYYY:MM:DD) and ect.

*flags* Set of flags: OSANY|OSCENTURY|OSYEAR|OSMONTH|OSWEEK|OSDAY.

#### Returns

Completion status of operation:

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

### 6.5.3.20 int pe\_DateTimeStr (OSCTXT \* *pctxt*, const char \* *string*, OSUINT32 *flags*)

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

#### Parameters

*pctxt* Pointer to context block structure.

*string* Character string variable containing value to be encoded in string form (YYYY-MM-DDTHH:MM:SS) and ect.

*flags* 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.

### 6.5.3.21 int pe\_Duration (OSCTXT \* *pctxt*, const char \* *string*, OSBOOL *rec*)

This function will encode an ISO 8601 DURATION types.

#### Parameters

*pctxt* Pointer to context block structure.

*string* Character string variable containing value to be encoded in string form (PnYnMnDTnHnMnS).

*rec* Encode recursive interval (Rn/).

#### Returns

Completion status of operation:

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

### 6.5.3.22 OSUINT32 pe\_GetIntLen (OSUINT32 *value*)

#### Parameters

*value* Length value to be encoded.

### 6.5.3.23 size\_t pe\_GetMsgBitCnt (OSCTXT \* *pctxt*)

#### Parameters

*pctxt* 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().

### 6.5.3.24 OSOCTET\* pe\_GetMsgPtr (OSCTXT \* *pctxt*, 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

*pctxt* 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.

*pLength* Pointer to variable to receive length of the encoded message.

#### Returns

Completion status of operation:

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

### 6.5.3.25 OSOCTET\* pe\_GetMsgPtr64 (OSCTXT \* *pctxt*, OSSIZE \* *pLength*)

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

See also

[pe\\_GetMsgPtr](#)

### 6.5.3.26 OSOCTET\* 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](#)

### 6.5.3.27 int pe\_Interval (OSCTXT \* *pctxt*, const char \* *string*, OSBOOL *rec*, OSUINT32 *startFlags*, OSUINT32 *endFlags*)

This function will encode an ISO 8601 INTERVAL type.

#### Parameters

*pctxt* Pointer to context block structure.

*string* Character string variable containing value to be encoded in string form (start/end).

*rec* Encode recursive interval (Rn/).

*startFlags* 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.

*endFlags* Set format flags of interval end.

#### Returns

Completion status of operation:

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

### 6.5.3.28 int pe\_Length (OSCTXT \* *pctxt*, OSSIZE *value*)

This function will encode a length determinant value.

#### Parameters

*pctxt* 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.

*value* 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:  $16384 \leq \text{return} \leq 65536$  return  $\leq$  value
- value (the given length) Otherwise, a negative error status code.

#### 6.5.3.29 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

*pctxt* Pointer to context block structure.

*value* Unsigned integer value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.5.3.30 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

*pctxt* Pointer to context block structure.

*value* Unsigned 64-bit integer value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.5.3.31 int pe\_ObjectIdentifier (OSCTXT \* *pctxt*, ASN1OBJID \* *pvalue*)

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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.

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

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

#### Parameters

*pctxt* 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.

*numocts* Number of octets in the string to be encoded.

*data* Pointer to octet string data to be encoded.

#### Returns

Completion status of operation:

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

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

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

#### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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.

### 6.5.3.34 `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

*pctxt* 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.

*numocts* Number of octets in the string to be encoded.

*data* Pointer to octet string data to be encoded.

#### Returns

Completion status of operation:

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

### 6.5.3.35 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

*pctxt* 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.

*pos* Position that was saved in pe\_OpenTypeStart function.

*pPerField* 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.

### 6.5.3.36 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

*pctxt* 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.

*pElemList* A pointer to the open type to be encoded.

#### Returns

Completion status of operation:

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

### 6.5.3.37 int pe\_OpenTypeExtBits (OSCTXT \* *pctxt*, OSRTDList \* *pElemList*)

#### Parameters

*pctxt* 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.

*pElemList* 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 `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

*pctxt* 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.

*pPos* A pointer to return current buffer position.

*ppPerField* A pointer to to return current bit field record.

#### Returns

Completion status of operation:

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

### 6.5.3.39 `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

*pctxt* 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.

*value* Value to be encoded. Special real values plus and minus infinity are encoded by using the `rtxGetPlusInfinity` and the `rtxGetMinusInfinity` functions to se the real value to be encoded.

#### Returns

Completion status of operation:

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

### 6.5.3.40 `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

*pctxt* 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.

*pvalue* Value to be encoded.

#### Returns

Completion status of operation:

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

#### 6.5.3.41 int pe\_RelativeOID (OSCTXT \* *pctxt*, ASN1OBJID \* *pvalue*)

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

##### Parameters

*pctxt* Pointer to context block structure.

*pvalue* 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.

#### 6.5.3.42 int pe\_SemiConsInt64 (OSCTXT \* *pctxt*, OSINT64 *value*, OSINT64 *lower*)

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

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as 64-bit integer.

*lower* Lower range value, represented as signed 64-bit integer.

##### Returns

Completion status of operation:

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

#### 6.5.3.43 int pe\_SemiConsUInt64 (OSCTXT \* *pctxt*, OSUINT64 *value*, OSUINT64 *lower*)

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

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as unsigned 64-bit integer.

*lower* Lower range value, represented as unsigned 64-bit integer.

##### Returns

Completion status of operation:

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

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

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

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as unsigned 64-bit integer.

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

This function encodes an semi-constrained unsigned integer.

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

*lower* Lower range value, represented as unsigned integer.

##### Returns

Completion status of operation:

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

#### 6.5.3.46 `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

*pctxt* 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.

*value* An unsigned integer value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.5.3.47 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

*pctxt* 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.

*value* An unsigned integer value to be encoded.

##### Returns

Completion status of operation:

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

#### 6.5.3.48 int pe\_TimeStr (OSCTXT \* *pctxt*, const char \* *string*, OSUINT32 *flags*)

This function will encode an ISO 8601 TIME types.

##### Parameters

*pctxt* Pointer to context block structure.

*string* Character string variable containing value to be encoded in string form (HH:MM:SS) and ect.

*flags* 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.

#### 6.5.3.49 int pe\_UnconsInt64 (OSCTXT \* *pctxt*, OSINT64 *value*)

This function encodes an unconstrained 64-bit integer.

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as 64-bit integer.

##### Returns

Completion status of operation:

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

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

This function encodes an unconstrained integer.

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

##### Returns

Completion status of operation:

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

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

This function encodes an unconstrained length value.

##### Parameters

*pctxt* 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.

*value* 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.

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

This function encodes an unconstrained unsigned 64-bit integer.

##### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded, represented as unsigned 64-bit integer.

##### Returns

Completion status of operation:

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

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

This function encodes an unconstrained unsigned integer.



## Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

## Returns

Completion status of operation:

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

### 6.5.3.54 `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

*pctxt* 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.

*value* 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.

*permCharSet* 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.

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

This function will encode a ASN.1 character string.

## Parameters

*pctxt* 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.

*value* 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.

### 6.5.3.56 int pe\_YearInt (OSCTXT \* *pctxt*, OSINT32 *value*)

This function will encode an ISO 8601 YEAR type.

#### Parameters

*pctxt* Pointer to context block structure.

*value* Value to be encoded.

#### Returns

Completion status of operation:

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

### 6.5.3.57 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

*pctxt* Pointer to context block structure.

*string* Pointer to string to be encoded.

*charSet* String containing permitted alphabet character set. Can be null if no character set was specified.

*nbits* Number of bits in a character set character (unaligned).

*canSetBits* 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

### Defines

- #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 **pe\_resetp**(pctxt) rtxResetContext(pctxt)
- #define **pd\_resetp**(pctxt) rtxResetContext(pctxt)

### 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 **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)
- void **pu\_bindump** (OSCTXT \*pctxt, 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)

## 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 int pu\_addSizeConstraint (OSCTXT \*pctx, const Asn1SizeCnst \*pSize)

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

#### Parameters

*pctx* A pointer to a context structure. The referenced size constraint is added to this structure for use by a subsequent encode or decode function.

*pSize* A pointer to the size constraint to add the context variable.

#### Returns

Completion status of operation:

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

### 6.6.2.2 void pu\_bindump (OSCTXT \*pctx, 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

*pctx* A 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.

*varname* The name of the top-level variable name of the structure being dumped.

Referenced by ASN1PERMessageBuffer::binDump().

### 6.6.2.3 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

*pSizeCnst* A pointer to a size constraint structure.

*value* The value to be checked.

*pExtendable* Pointer to boolean indicating if size constraint is extensible.

*pSizeRange* Size range which value falls within.

*pExtSize* 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.

### 6.6.2.4 void pu\_freeContext (OSCTXT \* *pctxt*)

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

*pctxt* 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.

### 6.6.2.5 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

### 6.6.2.6 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.

### 6.6.2.7 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

*pctxt* 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().

#### 6.6.2.8 `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

*pctxt* 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.

#### 6.6.2.9 `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

*pctxt* 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().

#### 6.6.2.10 `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

*pctxt* 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.

*bufaddr* 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.

*bufsiz* 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.

*aligned* 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.

### 6.6.2.11 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

*pTarget* A pointer to the target context structure. Buffer information within this structure is updated with data from the source context.

*pSource* 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.

### 6.6.2.12 void pu\_initFieldList (OSCTXT \* *pctx*, OSINT16 *bitOffset*)

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

## Parameters

*pctx* A pointer to a context structure.

*bitOffset* Current bit offset.

### 6.6.2.13 void pu\_initRtxDiagBitFieldsList (OSCTXT \* *pctx*, 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

*pctx* A pointer to a context structure.

*bitOffset* Current bit offset.

#### 6.6.2.14 void pu\_insLenField (OSCTXT \* *pctxt*)

##### Parameters

*pctxt* 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.

#### 6.6.2.15 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

*pSizeList* A pointer to a size constraint structure.

##### Returns

TRUE if size constraint is fixed.

#### 6.6.2.16 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

*bufaddr* 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.

*bufsiz* 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.

*aligned* 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.

#### 6.6.2.17 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

*pctxt* A pointer to a context structure.

*nameSuffix* Suffix to be append to current element name.



#### 6.6.2.18 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

*pctxt* A pointer to a context structure.

#### 6.6.2.19 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

*pctxt* A pointer to a context structure.

*idx* The location to insert the element.

#### 6.6.2.20 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

*pctxt* A pointer to a context structure.

*name* A pointer to the element to add to the stack.

#### 6.6.2.21 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

*pctxt* Pointer to a context structure.

*pCharSet* 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.

*pAlphabet* Pointer to a structure describing the 16-bit permitted alphabet.

##### Returns

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

**6.6.2.22 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**

*pCharSet* 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.

*firstChar* The first character in the range.

*lastChar* The last character in the range.

**6.6.2.23 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**

*pctxt* Pointer to a context structure.

*pCharSet* 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.

*pAlphabet* Pointer to a structure describing the 32-bit permitted alphabet.

**Returns**

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

**6.6.2.24 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**

*pCharSet* 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.

*firstChar* The first character in the range.

*lastChar* The last character in the range.

**6.6.2.25 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

*pctxt* Pointer to a context structure.

*bufaddr* Address of memory buffer. For encoding, this may be set to NULL to indicate a dynamic buffer is to be used.

*bufsiz* Size, in byte, of static memory buffer.

*aligned* Indicate if aligned (true) or unaligned (false) PER should be used for encoding or decoding.

### 6.6.2.26 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

*pCharSet* 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.

*permSet* A null-terminated string of permitted characters.

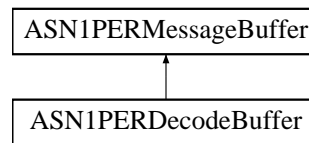
# Chapter 7

## 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, OSRTCContext \*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)

#### 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::ASN1PERDecodeBuffer (OSBOOL *aligned*) [inline]

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

#### Parameters

*aligned* Flag indicating if the message was encoded using aligned (TRUE)\* or unaligned (FALSE) encoding.

### 7.1.2.2 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

*pMsgBuf* A pointer to the message to be decoded.

*msgBufLen* Length of the message buffer.

*aligned* Flag indicating if the message was encoded using aligned (TRUE) \* or unaligned (FALSE) encoding.

### 7.1.2.3 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

*pMsgBuf* A pointer to the message to be decoded.

*msgBufLen* Length of the message buffer.

*aligned* Flag indicating if the message was encoded using aligned (TRUE) \* or unaligned (FALSE) encoding.

*pContext* A pointer to an OSRTContext structure created by the user.

### 7.1.2.4 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

*istream* A reference to an input stream object.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

#### 7.1.2.5 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

*filePath* Complete file path and name of file to read.

*aligned* Flag indicating if the message was encoded using aligned (TRUE) \* or unaligned (FALSE) encoding.

### 7.1.3 Member Function Documentation

#### 7.1.3.1 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.

References pd\_byte\_align().

#### 7.1.3.2 virtual OSBOOL ASN1PERDecodeBuffer::isA (Type *bufferType*) [inline, virtual]

This method checks the type of the message buffer.

##### Parameters

*bufferType* 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 EXTPERMETHOD 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

*ub* Single byte buffer to receive peeked byte.

##### Returns

Completion status of operation:

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

#### 7.1.3.4 EXTPERMETHOD int ASN1PERDecodeBuffer::readBinaryFile (const char \* *filePath*)

This method reads the file into the buffer to decode.

##### Parameters

*filePath* 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 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

*buffer* Buffer into which data should be read.

*bufsize* Size of the buffer

*nbytes* Number of bytes to read. Must be <= bufsize.

##### Returns

Completion status of operation:

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

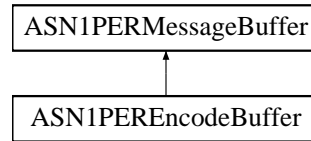
The documentation for this class was generated from the following file:

- [asn1PerCppType.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)
- EXTPERMETHOD [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](#) ()

### 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::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

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.



### 7.2.2.2 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

*pMsgBuf* A pointer to a fixed-size message buffer to receive the encoded message.

*msgBufLen* Size of the fixed-size message buffer.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

### 7.2.2.3 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

*pMsgBuf* A pointer to a fixed-size message buffer to receive the encoded message.

*msgBufLen* Size of the fixed-size message buffer.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

*pContext* A pointer to an OSRTContext structure created by the user.

### 7.2.2.4 EXTPERMETHOD 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

*ostream* A reference to an output stream object.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

## 7.2.3 Member Function Documentation

### 7.2.3.1 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 int ASN1PEREncodeBuffer::encodeBit (OSBOOL *value*) [inline]

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

#### Parameters

*value* Boolean value of bit to be written.

#### Returns

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

### 7.2.3.3 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

*pvalue* Pointer to byte array containing data to be encoded.

*nbits* Number of bits to copy from byte array to encode buffer.

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

This method checks the type of the message buffer.

#### **Parameters**

*bufferType* 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.

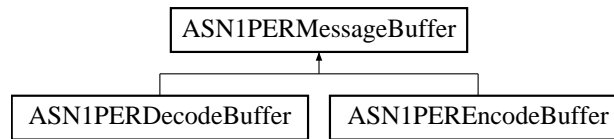
The documentation for this class was generated from the following file:

- [asn1PerCppType.h](#)

## 7.3 ASN1PERMessageBuffer Class Reference

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

Inheritance diagram for ASN1PERMessageBuffer:



### Public Member Functions

- void `binDump` (const char \*varname)
- void `hexDump` ()
- 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, OSBOOL aligned)
- EXTPERMETHOD `ASN1PERMessageBuffer` (Type *bufferType*, OSOCTET \*pMsgBuf, size\_t msgBufLen, OSBOOL 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 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

*bufferType* Type of message buffer that is being created (for example, PEREncode or PERDecode).

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

### 7.3.2.2 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

*stream* Stream class reference.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

### 7.3.2.3 EXTPERMETHOD ASN1PERMessageBuffer::ASN1PERMessageBuffer (Type *bufferType*, OSOCTET \* *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

*bufferType* Type of message buffer that is being created (for example, PEREncode or PERDecode).

*pMsgBuf* A pointer to a fixed size message buffer to receive the encoded message.

*msgBufLen* Size of the fixed-size message buffer.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

### 7.3.2.4 EXTPERMETHOD ASN1PERMessageBuffer::ASN1PERMessageBuffer (Type *bufferType*, OSOCTET \* *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

*bufferType* Type of message buffer that is being created (for example, PEREncode or PERDecode).

*pMsgBuf* A pointer to a fixed size message buffer to receive the encoded message.

*msgBufLen* Size of the fixed-size message buffer.

*aligned* Flag indicating if aligned (TRUE) or unaligned (FALSE) encoding should be done.

*pContext* A pointer to an OSRTContext structure.

### 7.3.3 Member Function Documentation

#### 7.3.3.1 void ASN1PERMessageBuffer::binDump (const char \* *varname*) [inline]

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

##### Parameters

*varname* char pointer to current buffer

References pu\_bindump().

#### 7.3.3.2 size\_t ASN1PERMessageBuffer::getMsgLen () [inline]

This method returns the length of a previously encoded PER message.

##### Parameters

- none

References pu\_getMsgLen().

#### 7.3.3.3 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 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 EXTPERMETHOD int ASN1PERMessageBuffer::setBuffer (const OSOCTET \* *pMsgBuf*, size\_t *msgBufLen*)

This method sets a buffer to receive the encoded message.

##### Parameters

*pMsgBuf* 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).

*msgBufLen* 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 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

*value* 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](#)

#### Defines

- #define **ASN\_K\_EXTENUM** 999
- #define **MAX\_BIGINTBYTES** (1024)
- #define **OSYEAR\_BASIC** OSUINCONST(0x8000000)
- #define **OSYEAR\_PROLEPTIC** OSUINCONST(0x4000000)
- #define **OSYEAR\_NEGATIVE** OSUINCONST(0x2000000)
- #define **OSYEAR\_L(n)** ((OSUINCONST32)(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\_CHECKSEQOFLN**(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 **pe\_resetp**(pctxt) rtxResetContext(pctxt)
- #define **pd\_resetp**(pctxt) rtxResetContext(pctxt)
- #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|OSUTC|(n))
- #define **pe\_MinutesAndDiffAndFraction**(pctxt, string, n) pe\_TimeStr (pctxt, string, OSHOURS|OSMINUTES|OSDIFF|(n))
- #define **pe\_RecStartDateInterval**(pctxt, string, flags) pe\_Interval (pctxt, string, TRUE, flags, flags)
- #define **pe\_RecStartTimeInterval**(pctxt, string, flags) pe\_Interval (pctxt, string, TRUE, flags, flags)
- #define **pe\_RecStartDateTimeInterval**(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, OSDURATION)
- #define **pe\_RecStartTimeDurationInterval**(pctxt, string, flags) pe\_Interval (pctxt, string, TRUE, flags, OSDURATION)
- #define **pe\_RecStartDateTimeDurationInterval**(pctxt, string, flags) pe\_Interval (pctxt, string, TRUE, flags, OSDURATION)
- #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\\_ConsInteger](#) (OSCTXT \*pctxt, OSINT32 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsInt8](#) (OSCTXT \*pctxt, OSINT8 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsInt16](#) (OSCTXT \*pctxt, OSINT16 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsInt64](#) (OSCTXT \*pctxt, OSINT64 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsUnsigned](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd\\_ConsUnsignedSignedBound](#) (OSCTXT \*pctxt, OSUINT32 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsUInt8](#) (OSCTXT \*pctxt, OSUINT8 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd\\_ConsUInt8SignedBound](#) (OSCTXT \*pctxt, OSUINT8 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsUInt16](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd\\_ConsUInt16SignedBound](#) (OSCTXT \*pctxt, OSUINT16 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsUInt64](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue, OSUINT64 lower, OSUINT64 upper)
- int [pd\\_ConsUInt64SignedBound](#) (OSCTXT \*pctxt, OSUINT64 \*pvalue, OSINT64 lower, OSINT64 upper)
- int [pd\\_ConsWholeNumber](#) (OSCTXT \*pctxt, OSUINT32 \*padjusted\_value, OSUINT32 range\_value)
- int [pd\\_ConsWholeNumber64](#) (OSCTXT \*pctxt, OSUINT64 \*padjusted\_value, OSUINT64 range\_value)
- int [pd\\_ConstrainedString](#) (OSCTXT \*pctxt, const char \*\*string, Asn1CharSet \*pCharSet)
- int [pd\\_ConstrainedStringEx](#) (OSCTXT \*pctxt, const char \*\*string, const char \*charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int [pd\\_ConstrFixedLenStringEx](#) (OSCTXT \*pctxt, char \*strbuf, size\_t bufsiz, const char \*charSet, OSUINT32 abits, OSUINT32 ubits, OSUINT32 canSetBits)
- int [pd\\_16BitConstrainedString](#) (OSCTXT \*pctxt, Asn116BitCharString \*pString, Asn116BitCharSet \*pCharSet)
- int [pd\\_32BitConstrainedString](#) (OSCTXT \*pctxt, Asn132BitCharString \*pString, Asn132BitCharSet \*pCharSet)
- int [pd\\_DateStr](#) (OSCTXT \*pctxt, const char \*\*string, OSUINT32 flags)
- int [pd\\_DateTimeStr](#) (OSCTXT \*pctxt, const char \*\*string, OSUINT32 flags)
- int [pd\\_Duration](#) (OSCTXT \*pctxt, const char \*\*string, OSBOOL rec)
- int [pd\\_DynBitString](#) (OSCTXT \*pctxt, ASN1DynBitStr \*pBitStr)
- int [pd\\_DynBitString64](#) (OSCTXT \*pctxt, ASN1DynBitStr64 \*pBitStr)
- int [pd\\_DynOctetString](#) (OSCTXT \*pctxt, ASN1DynOctStr \*pOctStr)
- int [pd\\_DynOctetString64](#) (OSCTXT \*pctxt, OSDynOctStr64 \*pOctStr)
- int [pd\\_GetBinStrDataOffset](#) (OSCTXT \*pctxt, OSUINT32 \*pnumbits, OSBOOL bitStrFlag)
- int [pd\\_GetComponentLength](#) (OSCTXT \*pctxt, OSUINT32 itemBits)
- int [pd\\_GetComponentLength64](#) (OSCTXT \*pctxt, 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 \*\*pvalue)
- 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 numocts, const OSOCTET \*data)
- int [pe\\_BitString32](#) (OSCTXT \*pctxt, ASN1BitStr32 \*pbitstr, OSUINT32 lower, OSUINT32 upper)
- 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\\_ConsInt64](#) (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 \*pctxt)
- OSOCTET \* `pe_GetMsgPtr` (OSCTXT \*pctxt, OSINT32 \*pLength)
- OSOCTET \* `pe_GetMsgPtrU` (OSCTXT \*pctxt, OSUINT32 \*pLength)
- OSOCTET \* `pe_GetMsgPtr64` (OSCTXT \*pctxt, OSSIZE \*pLength)
- int `pe_Interval` (OSCTXT \*pctxt, const char \*string, OSBOOL rec, OSUINT32 startFlags, OSUINT32 endFlags)
- int `pe_Length` (OSCTXT \*pctxt, OSSIZE value)
- int `pe_NonNegBinInt` (OSCTXT \*pctxt, OSUINT32 value)
- int `pe_NonNegBinInt64` (OSCTXT \*pctxt, OSUINT64 value)
- int `pe_ObjectIdentifier` (OSCTXT \*pctxt, ASN1OBJID \*pvalue)
- int `pe_oid64` (OSCTXT \*pctxt, ASN1OID64 \*pvalue)
- int `pe_RelativeOID` (OSCTXT \*pctxt, ASN1OBJID \*pvalue)
- int `pe-OctetString` (OSCTXT \*pctxt, OSSIZE numocts, const OSOCTET \*data)
- int `pe_OpenType` (OSCTXT \*pctxt, OSSIZE numocts, const OSOCTET \*data)
- int `pe_OpenTypeEnd` (OSCTXT \*pctxt, OSUINT32 pos, void \*pPerField)
- int `pe_OpenTypeExt` (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int `pe_OpenTypeExtBits` (OSCTXT \*pctxt, OSRTDList \*pElemList)
- int `pe_OpenTypeStart` (OSCTXT \*pctxt, OSUINT32 \*pPos, void \*\*ppPerField)
- int `pe_Real` (OSCTXT \*pctxt, OSREAL value)
- int `pe_SmallNonNegWholeNumber` (OSCTXT \*pctxt, OSUINT32 value)
- int `pe_SmallLength` (OSCTXT \*pctxt, OSSIZE value)
- int `pe_SemiConsInt64` (OSCTXT \*pctxt, OSINT64 value, OSINT64 lower)
- int `pe_SemiConsUnsignedSignedBound` (OSCTXT \*pctxt, OSUINT32 value, OSINT64 lower)
- int `pe_SemiConsUInt64` (OSCTXT \*pctxt, OSUINT64 value, OSUINT64 lower)
- int `pe_SemiConsUInt64SignedBound` (OSCTXT \*pctxt, OSUINT64 value, OSINT64 lower)
- int `pe_TimeStr` (OSCTXT \*pctxt, const char \*string, OSUINT32 flags)
- int `pe_UnconsLength` (OSCTXT \*pctxt, OSSIZE value)
- int `pe_UnconsInteger` (OSCTXT \*pctxt, OSINT32 value)
- int `pe_UnconsInt64` (OSCTXT \*pctxt, OSINT64 value)
- int `pe_UnconsUnsigned` (OSCTXT \*pctxt, OSUINT32 value)
- int `pe_UnconsUInt64` (OSCTXT \*pctxt, OSUINT64 value)
- int `pe_VarWidthCharString` (OSCTXT \*pctxt, const char \*value)
- int `pe_YearInt` (OSCTXT \*pctxt, OSINT32 value)
- int `pe_Real10` (OSCTXT \*pctxt, const char \*pvalue)
- int `uperEncConstrString` (OSCTXT \*pctxt, const char \*string, const char \*charSet, OSUINT32 nbits, OSUINT32 canSetBits)
- 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 *pctx`)
- `size_t pu_getMsgLenBits` (`OSCTXT *pctx`)
- `void pu_hexdump` (`OSCTXT *pctx`)
- `int pu_setBuffer` (`OSCTXT *pctx`, `OSOCKET *bufaddr`, `size_t bufsiz`, `OSBOOL aligned`)
- `int pu_initContext` (`OSCTXT *pctx`, `OSOCKET *bufaddr`, `OSUIN32 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` (`OSOCKET *bufaddr`, `OSUIN32 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`, `OSUIN16 firstChar`, `OSUIN16 lastChar`)
- `void pu_setFldBitCount` (`OSCTXT *pctx`)
- `void pu_setFldBitOffset` (`OSCTXT *pctx`)
- `void pu_setFldListFromCtxt` (`OSCTXT *pctx`, `OSCTXT *srcctx`)
- `void pu_setOpenTypeFldList` (`OSCTXT *pctx`, `OSRTSList *plist`)
- `void pu_setRtxDiagOpenTypeFldList` (`OSRTDiagBitFieldList *pMainBFList`, `OSRTDiagBitFieldList *pOpenTypeBFList`)
- `OSBOOL pu_setTrace` (`OSCTXT *pCtxt`, `OSBOOL value`)
- `void pu_setAligned` (`OSCTXT *pctx`, `OSBOOL value`)
- `void pu_deleteFieldList` (`OSCTXT *pctx`)
- `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`, `OSUIN32 firstChar`, `OSUIN32 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

- asn1per.h, 94
- asn1PerCppTypes.h, 104
- ASN1PERDecodeBuffer, 80
  - ASN1PERDecodeBuffer, 81
  - byteAlign, 82
  - isA, 82
  - peekByte, 82
  - readBinaryFile, 82
  - readBytes, 83
- ASN1PEREncodeBuffer, 84
  - ASN1PEREncodeBuffer, 84, 85
  - byteAlign, 85
  - encodeBit, 85
  - encodeBits, 86
  - getMsgBitCnt, 86
  - getMsgCopy, 86
  - getMsgPtr, 86
  - init, 87
  - isA, 87
- ASN1PERMessageBuffer, 88
  - ASN1PERMessageBuffer, 88, 89
  - binDump, 90
  - getMsgLen, 90
  - hexDump, 90
  - isAligned, 90
  - setBuffer, 90
  - setTrace, 91
- binDump
  - ASN1PERMessageBuffer, 90
- BinDumpBuffer, 92
- byteAlign
  - ASN1PERDecodeBuffer, 82
  - ASN1PEREncodeBuffer, 85
- encodeBit
  - ASN1PEREncodeBuffer, 85
- encodeBits
  - ASN1PEREncodeBuffer, 86
- getMsgBitCnt
  - ASN1PEREncodeBuffer, 86
- getMsgCopy
  - ASN1PEREncodeBuffer, 86
- getMsgLen
  - ASN1PERMessageBuffer, 90
- getMsgPtr
  - ASN1PEREncodeBuffer, 86
- hexDump
  - ASN1PERMessageBuffer, 90
- init
  - ASN1PEREncodeBuffer, 87
- isA
  - ASN1PERDecodeBuffer, 82
  - ASN1PEREncodeBuffer, 87
- isAligned
  - ASN1PERMessageBuffer, 90
- pd\_16BitConstrainedString
  - perdecruntime, 18
- pd\_32BitConstrainedString
  - perdecruntime, 18
- pd\_BigInteger
  - perdecruntime, 19
- pd\_BigIntegerEx
  - perdecruntime, 19
- pd\_BigIntegerValue
  - perdecruntime, 19
- pd\_bit
  - perruntime, 13
- pd\_BitString
  - perdecruntime, 20
- pd\_BitString32
  - perdecruntime, 20
- pd\_BitString64
  - perdecruntime, 20
- pd\_BMPString
  - perdecruntime, 21
- pd\_byte\_align
  - perdecruntime, 21
- PD\_BYTE\_ALIGN0
  - perruntime, 13
- PD\_CHECKSEQOFLN
  - perruntime, 13
- pd\_ChoiceOpenTypeExt
  - perdecruntime, 21
- pd\_ConsInt16
  - perdecruntime, 22
- pd\_ConsInt64
  - perdecruntime, 22

pd\_ConstInt8  
     perdecruntime, 22  
 pd\_ConstInteger  
     perdecruntime, 23  
 pd\_ConstrainedString  
     perdecruntime, 23  
 pd\_ConstrainedStringEx  
     perdecruntime, 23  
 pd\_ConstrFixedLenStringEx  
     perdecruntime, 24  
 pd\_ConstUInt16  
     perdecruntime, 24  
 pd\_ConstUInt16SignedBound  
     perdecruntime, 25  
 pd\_ConstUInt64  
     perdecruntime, 25  
 pd\_ConstUInt64SignedBound  
     perdecruntime, 25  
 pd\_ConstUInt8  
     perdecruntime, 26  
 pd\_ConstUInt8SignedBound  
     perdecruntime, 26  
 pd\_ConstUnsigned  
     perdecruntime, 26  
 pd\_ConstUnsignedSignedBound  
     perdecruntime, 27  
 pd\_ConstWholeNumber  
     perdecruntime, 27  
 pd\_ConstWholeNumber64  
     perdecruntime, 27  
 pd\_DateStr  
     perdecruntime, 28  
 pd\_DateTime  
     perruntime, 13  
 pd\_DateTimeStr  
     perdecruntime, 28  
 pd\_Duration  
     perdecruntime, 28  
 pd\_DynBitString  
     perdecruntime, 29  
 pd\_DynBitString64  
     perdecruntime, 29  
 pd\_DynOctetString  
     perdecruntime, 29  
 pd\_DynOctetString64  
     perdecruntime, 30  
 pd\_GetBinStrDataOffset  
     perdecruntime, 30  
 pd\_GetComponentLength  
     perdecruntime, 30  
 pd\_GetComponentLength64  
     perdecruntime, 31  
 pd\_Interval  
     perdecruntime, 31  
 pd\_isFragmented  
     perdecruntime, 31  
 pd\_Length  
     perdecruntime, 32  
 pd\_Length64  
     perdecruntime, 32  
 pd\_moveBitCursor  
     perdecruntime, 18  
 pd\_NumericString  
     perruntime, 14  
 pd\_ObjectIdentifier  
     perdecruntime, 32  
 pd\_OctetString  
     perdecruntime, 33  
 pd\_OctetString64  
     perdecruntime, 33  
 pd\_oid64  
     perdecruntime, 33  
 pd\_OpenType  
     perdecruntime, 34  
 pd\_OpenTypeExt  
     perdecruntime, 34  
 pd\_Real  
     perdecruntime, 35  
 pd\_Real10  
     perdecruntime, 35  
 pd\_RelativeOID  
     perdecruntime, 35  
 pd\_SemiConstInt16  
     perdecruntime, 36  
 pd\_SemiConstInt64  
     perdecruntime, 36  
 pd\_SemiConstInt8  
     perdecruntime, 36  
 pd\_SemiConstInteger  
     perdecruntime, 36  
 pd\_SemiConstUInt16  
     perdecruntime, 37  
 pd\_SemiConstUInt16SignedBound  
     perdecruntime, 37  
 pd\_SemiConstUInt64  
     perdecruntime, 37  
 pd\_SemiConstUInt64SignedBound  
     perdecruntime, 38  
 pd\_SemiConstUInt8  
     perdecruntime, 38  
 pd\_SemiConstUInt8SignedBound  
     perdecruntime, 38  
 pd\_SemiConstUnsigned  
     perdecruntime, 39  
 pd\_SemiConstUnsignedSignedBound  
     perdecruntime, 39  
 pd\_SmallLength  
     perdecruntime, 39

pd\_SmallNonNegWholeNumber  
     perdecruntime, 40  
 pd\_TimeStr  
     perdecruntime, 40  
 pd\_UnconsInt16  
     perdecruntime, 40  
 pd\_UnconsInt64  
     perdecruntime, 41  
 pd\_UnconsInt8  
     perdecruntime, 41  
 pd\_UnconsInteger  
     perdecruntime, 41  
 pd\_UnconsLength  
     perdecruntime, 42  
 pd\_UnconsLength64  
     perdecruntime, 42  
 pd\_UnconsUInt16  
     perdecruntime, 42  
 pd\_UnconsUInt64  
     perdecruntime, 43  
 pd\_UnconsUInt8  
     perdecruntime, 43  
 pd\_UnconsUnsigned  
     perdecruntime, 43  
 pd\_UniversalString  
     perdecruntime, 44  
 pd\_VarWidthCharString  
     perdecruntime, 44  
 pd\_YearInt  
     perdecruntime, 44  
 pe\_16BitConstrainedString  
     perencruntime, 52  
 pe\_2sCompBinInt  
     perencruntime, 52  
 pe\_2sCompBinInt64  
     perencruntime, 52  
 pe\_32BitConstrainedString  
     perencruntime, 52  
 pe\_aligned\_octets  
     perencruntime, 53  
 pe\_BigInteger  
     perencruntime, 53  
 pe\_bit  
     perencruntime, 48  
 pe\_bits  
     perencruntime, 49  
 pe\_bits64  
     perencruntime, 53  
 pe\_BitString  
     perencruntime, 54  
 pe\_BMPString  
     perencruntime, 54  
 pe\_byte\_align  
     perencruntime, 54  
 pe\_CheckBuffer  
     perencruntime, 49  
 pe\_ChoiceTypeExt  
     perencruntime, 55  
 pe\_ConsInt64  
     perencruntime, 55  
 pe\_ConsInteger  
     perencruntime, 49  
 pe\_ConstrainedString  
     perencruntime, 55  
 pe\_ConstrainedStringEx  
     perencruntime, 56  
 pe\_ConsUInt64  
     perencruntime, 56  
 pe\_ConsUInt64SignedBound  
     perencruntime, 57  
 pe\_ConsUnsigned  
     perencruntime, 50  
 pe\_ConsUnsignedSignedBound  
     perencruntime, 50  
 pe\_ConsWholeNumber  
     perencruntime, 57  
 pe\_ConsWholeNumber64  
     perencruntime, 57  
 pe\_DateStr  
     perencruntime, 58  
 pe\_DateTime  
     perruntime, 14  
 pe\_DateTimeStr  
     perencruntime, 58  
 pe\_Duration  
     perencruntime, 58  
 pe\_GetIntLen  
     perencruntime, 59  
 pe\_GetMsgBitCnt  
     perencruntime, 59  
 pe\_GetMsgPtr  
     perencruntime, 59  
 pe\_GetMsgPtr64  
     perencruntime, 59  
 pe\_GetMsgPtrU  
     perencruntime, 60  
 pe\_Interval  
     perencruntime, 60  
 pe\_Length  
     perencruntime, 60  
 pe\_NonNegBinInt  
     perencruntime, 61  
 pe\_NonNegBinInt64  
     perencruntime, 61  
 pe\_NumericString  
     perruntime, 14  
 pe\_ObjectIdentifier  
     perencruntime, 61



- pe\_octets
  - perencruntime, 50
- pe\_OctetString
  - perencruntime, 61
- pe\_oid64
  - perencruntime, 62
- pe\_OpenType
  - perencruntime, 62
- pe\_OpenTypeEnd
  - perencruntime, 62
- pe\_OpenTypeExt
  - perencruntime, 63
- pe\_OpenTypeExtBits
  - perencruntime, 63
- pe\_OpenTypeStart
  - perencruntime, 63
- pe\_Real
  - perencruntime, 64
- pe\_Real10
  - perencruntime, 64
- pe\_RelativeOID
  - perencruntime, 64
- pe\_SemiConsInt64
  - perencruntime, 65
- pe\_SemiConsInteger
  - perencruntime, 51
- pe\_SemiConsUInt64
  - perencruntime, 65
- pe\_SemiConsUInt64SignedBound
  - perencruntime, 65
- pe\_SemiConsUnsigned
  - perencruntime, 51
- pe\_SemiConsUnsignedSignedBound
  - perencruntime, 66
- pe\_SmallLength
  - perencruntime, 66
- pe\_SmallNonNegWholeNumber
  - perencruntime, 66
- pe\_TimeStr
  - perencruntime, 67
- pe\_UnconsInt64
  - perencruntime, 67
- pe\_UnconsInteger
  - perencruntime, 67
- pe\_UnconsLength
  - perencruntime, 68
- pe\_UnconsUInt64
  - perencruntime, 68
- pe\_UnconsUnsigned
  - perencruntime, 68
- pe\_UniversalString
  - perencruntime, 69
- pe\_VarWidthCharString
  - perencruntime, 69
- pe\_YearInt
  - perencruntime, 69
- peekByte
  - ASN1PERDecodeBuffer, 82
- PER C Decode Functions., 16
- PER C Encode Functions., 47
- PER C Utility Functions, 71
- PER C++ Runtime Classes., 6
- PER Message Buffer Classes, 7
- PER Runtime Library Functions., 8
- perdecruntime
  - pd\_16BitConstrainedString, 18
  - pd\_32BitConstrainedString, 18
  - pd\_BigInteger, 19
  - pd\_BigIntegerEx, 19
  - pd\_BigIntegerValue, 19
  - pd\_BitString, 20
  - pd\_BitString32, 20
  - pd\_BitString64, 20
  - pd\_BMPString, 21
  - pd\_byte\_align, 21
  - pd\_ChoiceOpenTypeExt, 21
  - pd\_ConsInt16, 22
  - pd\_ConsInt64, 22
  - pd\_ConsInt8, 22
  - pd\_ConsInteger, 23
  - pd\_ConstrainedString, 23
  - pd\_ConstrainedStringEx, 23
  - pd\_ConstrFixedLenStringEx, 24
  - pd\_ConsUInt16, 24
  - pd\_ConsUInt16SignedBound, 25
  - pd\_ConsUInt64, 25
  - pd\_ConsUInt64SignedBound, 25
  - pd\_ConsUInt8, 26
  - pd\_ConsUInt8SignedBound, 26
  - pd\_ConsUnsigned, 26
  - pd\_ConsUnsignedSignedBound, 27
  - pd\_ConsWholeNumber, 27
  - pd\_ConsWholeNumber64, 27
  - pd\_DateStr, 28
  - pd\_DateTimeStr, 28
  - pd\_Duration, 28
  - pd\_DynBitString, 29
  - pd\_DynBitString64, 29
  - pd\_DynOctetString, 29
  - pd\_DynOctetString64, 30
  - pd\_GetBinStrDataOffset, 30
  - pd\_GetComponentLength, 30
  - pd\_GetComponentLength64, 31
  - pd\_Interval, 31
  - pd\_isFragmented, 31
  - pd\_Length, 32
  - pd\_Length64, 32
  - pd\_moveBitCursor, 18



pd\_ObjectIdentifier, 32  
 pd\_OctetString, 33  
 pd\_OctetString64, 33  
 pd\_oid64, 33  
 pd\_OpenType, 34  
 pd\_OpenTypeExt, 34  
 pd\_Real, 35  
 pd\_Real10, 35  
 pd\_RelativeOID, 35  
 pd\_SemiConsInt16, 36  
 pd\_SemiConsInt64, 36  
 pd\_SemiConsInt8, 36  
 pd\_SemiConsInteger, 36  
 pd\_SemiConsUInt16, 37  
 pd\_SemiConsUInt16SignedBound, 37  
 pd\_SemiConsUInt64, 37  
 pd\_SemiConsUInt64SignedBound, 38  
 pd\_SemiConsUInt8, 38  
 pd\_SemiConsUInt8SignedBound, 38  
 pd\_SemiConsUnsigned, 39  
 pd\_SemiConsUnsignedSignedBound, 39  
 pd\_SmallLength, 39  
 pd\_SmallNonNegWholeNumber, 40  
 pd\_TimeStr, 40  
 pd\_UnconsInt16, 40  
 pd\_UnconsInt64, 41  
 pd\_UnconsInt8, 41  
 pd\_UnconsInteger, 41  
 pd\_UnconsLength, 42  
 pd\_UnconsLength64, 42  
 pd\_UnconsUInt16, 42  
 pd\_UnconsUInt64, 43  
 pd\_UnconsUInt8, 43  
 pd\_UnconsUnsigned, 43  
 pd\_UniversalString, 44  
 pd\_VarWidthCharString, 44  
 pd\_YearInt, 44  
 uperDecConstrFixedLenString, 45  
 uperDecConstrString, 45  
 perencruntime  
   pe\_16BitConstrainedString, 52  
   pe\_2sCompBinInt, 52  
   pe\_2sCompBinInt64, 52  
   pe\_32BitConstrainedString, 52  
   pe\_aligned\_octets, 53  
   pe\_BigInteger, 53  
   pe\_bit, 48  
   pe\_bits, 49  
   pe\_bits64, 53  
   pe\_BitString, 54  
   pe\_BMPString, 54  
   pe\_byte\_align, 54  
   pe\_CheckBuffer, 49  
   pe\_ChoiceTypeExt, 55  
   pe\_ConsInt64, 55  
   pe\_ConsInteger, 49  
   pe\_ConstrainedString, 55  
   pe\_ConstrainedStringEx, 56  
   pe\_ConsUInt64, 56  
   pe\_ConsUInt64SignedBound, 57  
   pe\_ConsUnsigned, 50  
   pe\_ConsUnsignedSignedBound, 50  
   pe\_ConsWholeNumber, 57  
   pe\_ConsWholeNumber64, 57  
   pe\_DateStr, 58  
   pe\_DateTimeStr, 58  
   pe\_Duration, 58  
   pe\_GetIntLen, 59  
   pe\_GetMsgBitCnt, 59  
   pe\_GetMsgPtr, 59  
   pe\_GetMsgPtr64, 59  
   pe\_GetMsgPtrU, 60  
   pe\_Interval, 60  
   pe\_Length, 60  
   pe\_NonNegBinInt, 61  
   pe\_NonNegBinInt64, 61  
   pe\_ObjectIdentifier, 61  
   pe\_octets, 50  
   pe\_OctetString, 61  
   pe\_oid64, 62  
   pe\_OpenType, 62  
   pe\_OpenTypeEnd, 62  
   pe\_OpenTypeExt, 63  
   pe\_OpenTypeExtBits, 63  
   pe\_OpenTypeStart, 63  
   pe\_Real, 64  
   pe\_Real10, 64  
   pe\_RelativeOID, 64  
   pe\_SemiConsInt64, 65  
   pe\_SemiConsInteger, 51  
   pe\_SemiConsUInt64, 65  
   pe\_SemiConsUInt64SignedBound, 65  
   pe\_SemiConsUnsigned, 51  
   pe\_SemiConsUnsignedSignedBound, 66  
   pe\_SmallLength, 66  
   pe\_SmallNonNegWholeNumber, 66  
   pe\_TimeStr, 67  
   pe\_UnconsInt64, 67  
   pe\_UnconsInteger, 67  
   pe\_UnconsLength, 68  
   pe\_UnconsUInt64, 68  
   pe\_UnconsUnsigned, 68  
   pe\_UniversalString, 69  
   pe\_VarWidthCharString, 69  
   pe\_YearInt, 69  
   uperEncConstrString, 70  
 PERField, 93  
 perruntime

- pd\_bit, 13
- PD\_BYTE\_ALIGN0, 13
- PD\_CHECKSEQOFLen, 13
- pd\_DateTime, 13
- pd\_NumericString, 14
- pe\_DateTime, 14
- pe\_NumericString, 14
- PU\_GETPADBITS, 14
- PU\_GETSIZECONSTRAINT, 14
- PU\_SETCHARSET, 14
- PU\_SETCTXTBITOFFSET, 14
- PU\_SETSIZECONSTRAINT, 15
- perutil
  - pu\_addSizeConstraint, 72
  - pu\_bindump, 72
  - pu\_checkSizeExt, 72
  - pu\_freeContext, 73
  - pu\_GetLibInfo, 73
  - pu\_GetLibVersion, 73
  - pu\_getMsgLen, 73
  - pu\_getMsgLenBits, 74
  - pu\_hexdump, 74
  - pu\_initContext, 74
  - pu\_initContextBuffer, 75
  - pu\_initFieldList, 75
  - pu\_initRtxDiagBitFieldList, 75
  - pu\_insLenField, 75
  - pu\_isFixedSize, 76
  - pu\_newContext, 76
  - pu\_newField, 76
  - pu\_popName, 76
  - pu\_pushElemName, 77
  - pu\_pushName, 77
  - pu\_set16BitCharSet, 77
  - pu\_set16BitCharSetFromRange, 77
  - pu\_set32BitCharSet, 78
  - pu\_set32BitCharSetFromRange, 78
  - pu\_setBuffer, 78
  - pu\_setCharSet, 79
- pu\_addSizeConstraint
  - perutil, 72
- pu\_bindump
  - perutil, 72
- pu\_checkSizeExt
  - perutil, 72
- pu\_freeContext
  - perutil, 73
- pu\_GetLibInfo
  - perutil, 73
- pu\_GetLibVersion
  - perutil, 73
- pu\_getMsgLen
  - perutil, 73
- pu\_getMsgLenBits
  - perutil, 74
- PU\_GETPADBITS
  - perruntime, 14
- PU\_GETSIZECONSTRAINT
  - perruntime, 14
- pu\_hexdump
  - perutil, 74
- pu\_initContext
  - perutil, 74
- pu\_initContextBuffer
  - perutil, 75
- pu\_initFieldList
  - perutil, 75
- pu\_initRtxDiagBitFieldList
  - perutil, 75
- pu\_insLenField
  - perutil, 75
- pu\_isFixedSize
  - perutil, 76
- pu\_newContext
  - perutil, 76
- pu\_newField
  - perutil, 76
- pu\_popName
  - perutil, 76
- pu\_pushElemName
  - perutil, 77
- pu\_pushName
  - perutil, 77
- pu\_set16BitCharSet
  - perutil, 77
- pu\_set16BitCharSetFromRange
  - perutil, 77
- pu\_set32BitCharSet
  - perutil, 78
- pu\_set32BitCharSetFromRange
  - perutil, 78
- pu\_setBuffer
  - perutil, 78
- PU\_SETCHARSET
  - perruntime, 14
- pu\_setCharSet
  - perutil, 79
- PU\_SETCTXTBITOFFSET
  - perruntime, 14
- PU\_SETSIZECONSTRAINT
  - perruntime, 15
- readBinaryFile
  - ASN1PERDecodeBuffer, 82
- readBytes
  - ASN1PERDecodeBuffer, 83
- setBuffer

- ASN1PERMessageBuffer, [90](#)
- setTrace
  - ASN1PERMessageBuffer, [91](#)
- uperDecConstrFixedLenString
  - perdecruntime, [45](#)
- uperDecConstrString
  - perdecruntime, [45](#)
- uperEncConstrString
  - perencruntime, [70](#)