

ASN1C

ASN.1 Compiler
Version 7.1
C# Common Runtime Classes
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.

Contents

1	ASNIC C# Runtime Library	1
2	Namespace Documentation	3
2.1	Package Com.Objsys.Asn1.Runtime	3
2.1.1	Detailed Description	4
3	Class Documentation	7
3.1	Asn18BitCharString Class Reference	7
3.1.1	Detailed Description	8
3.1.2	Constructor & Destructor Documentation	8
3.1.2.1	Asn18BitCharString	8
3.1.2.2	Asn18BitCharString	8
3.1.3	Member Function Documentation	8
3.1.3.1	Decode	8
3.1.3.2	Decode	8
3.1.3.3	Decode	8
3.1.3.4	Decode	9
3.1.3.5	Decode	9
3.1.3.6	Encode	9
3.1.3.7	Encode	9
3.1.3.8	Encode	10
3.1.3.9	Encode	10
3.1.3.10	Encode	10
3.1.3.11	Encode	11
3.1.3.12	Encode	11
3.1.3.13	Encode	11
3.1.4	Member Data Documentation	11
3.1.4.1	BITSPERCHAR_A	11
3.1.4.2	BITSPERCHAR_U	12

3.2	Asn1BigInteger Class Reference	13
3.2.1	Detailed Description	14
3.2.2	Constructor & Destructor Documentation	14
3.2.2.1	Asn1BigInteger	14
3.2.2.2	Asn1BigInteger	14
3.2.2.3	Asn1BigInteger	14
3.2.2.4	Asn1BigInteger	14
3.2.3	Member Function Documentation	14
3.2.3.1	Decode	14
3.2.3.2	Decode	15
3.2.3.3	Decode	15
3.2.3.4	Decode	15
3.2.3.5	Decode	15
3.2.3.6	DecodeSigned	15
3.2.3.7	DecodeSigned	16
3.2.3.8	DecodeUnsigned	16
3.2.3.9	DecodeUnsigned	16
3.2.3.10	DecodeValue	16
3.2.3.11	DecodeXER	16
3.2.3.12	DecodeXML	16
3.2.3.13	Encode	17
3.2.3.14	Encode	17
3.2.3.15	Encode	17
3.2.3.16	Encode	17
3.2.3.17	Encode	18
3.2.3.18	Encode	18
3.2.3.19	Encode	18
3.2.3.20	Encode	18
3.2.3.21	Encode	19
3.2.3.22	EncodeAttribute	19
3.2.3.23	EncodeSigned	19
3.2.3.24	EncodeSigned	19
3.2.3.25	EncodeUnsigned	19
3.2.3.26	EncodeUnsigned	20
3.2.3.27	EncodeValue	20
3.2.3.28	Equals	20
3.2.3.29	Equals	20

3.2.3.30	GetHashCode	20
3.2.3.31	ToString	21
3.2.4	Member Data Documentation	21
3.2.4.1	_TAG	21
3.2.4.2	mValue	21
3.3	Asn1BitString Class Reference	22
3.3.1	Detailed Description	23
3.3.2	Member Enumeration Documentation	23
3.3.2.1	StringFormat	23
3.3.3	Constructor & Destructor Documentation	23
3.3.3.1	Asn1BitString	23
3.3.3.2	Asn1BitString	24
3.3.3.3	Asn1BitString	24
3.3.3.4	Asn1BitString	24
3.3.3.5	Asn1BitString	24
3.3.3.6	Asn1BitString	24
3.3.4	Member Function Documentation	25
3.3.4.1	Clear	25
3.3.4.2	Decode	25
3.3.4.3	Decode	25
3.3.4.4	Decode	25
3.3.4.5	Decode	25
3.3.4.6	Decode	26
3.3.4.7	Decode	26
3.3.4.8	DecodeContent	26
3.3.4.9	DecodeXER	26
3.3.4.10	DecodeXML	27
3.3.4.11	Encode	27
3.3.4.12	Encode	27
3.3.4.13	Encode	27
3.3.4.14	Encode	28
3.3.4.15	Encode	28
3.3.4.16	Encode	28
3.3.4.17	Encode	28
3.3.4.18	Encode	29
3.3.4.19	Encode	29
3.3.4.20	Encode	29

3.3.4.21	Encode	29
3.3.4.22	Encode	29
3.3.4.23	Encode	30
3.3.4.24	EncodeContent	30
3.3.4.25	Equals	30
3.3.4.26	Equals	30
3.3.4.27	Get	31
3.3.4.28	GetHashCode	31
3.3.4.29	GetOerEffectiveMin	31
3.3.4.30	IsNamedBitStr	31
3.3.4.31	Set	31
3.3.4.32	Set	32
3.3.4.33	ToBoolArray	32
3.3.4.34	ToHexString	32
3.3.4.35	toInputStream	32
3.3.4.36	ToString	32
3.3.5	Member Data Documentation	32
3.3.5.1	_TAG	32
3.3.5.2	mStringFormat	33
3.3.5.3	mValue	33
3.3.5.4	numbits	33
3.3.5.5	trimZeroBits	33
3.3.6	Property Documentation	33
3.3.6.1	Length	33
3.3.6.2	this	33
3.4	Asn1BMPSString Class Reference	34
3.4.1	Detailed Description	34
3.4.2	Constructor & Destructor Documentation	34
3.4.2.1	Asn1BMPSString	34
3.4.2.2	Asn1BMPSString	34
3.4.3	Member Function Documentation	35
3.4.3.1	Decode	35
3.4.3.2	Decode	35
3.4.3.3	Decode	35
3.4.3.4	Decode	35
3.4.3.5	Decode	36
3.4.3.6	Decode	36

3.4.3.7	Encode	36
3.4.3.8	Encode	37
3.4.3.9	Encode	37
3.4.3.10	Encode	37
3.4.3.11	Encode	38
3.4.3.12	Encode	38
3.4.3.13	Encode	38
3.4.3.14	Encode	38
3.4.3.15	Encode	38
3.4.3.16	Encode	39
3.4.4	Member Data Documentation	39
3.4.4.1	_TAG	39
3.4.4.2	BITSPERCHAR	39
3.5	Asn1Boolean Class Reference	40
3.5.1	Detailed Description	40
3.5.2	Constructor & Destructor Documentation	41
3.5.2.1	Asn1Boolean	41
3.5.2.2	Asn1Boolean	41
3.5.3	Member Function Documentation	41
3.5.3.1	Decode	41
3.5.3.2	Decode	41
3.5.3.3	Decode	41
3.5.3.4	Decode	41
3.5.3.5	DecodeXER	42
3.5.3.6	DecodeXML	42
3.5.3.7	Encode	42
3.5.3.8	Encode	42
3.5.3.9	Encode	43
3.5.3.10	Encode	43
3.5.3.11	Encode	43
3.5.3.12	Encode	43
3.5.3.13	Encode	44
3.5.3.14	Encode	44
3.5.3.15	Encode	44
3.5.3.16	Encode	44
3.5.3.17	EncodeAttribute	44
3.5.3.18	Equals	45

3.5.3.19	Equals	45
3.5.3.20	GetHashCode	45
3.5.3.21	setTrueEncodedByte	45
3.5.3.22	ToString	46
3.5.4	Member Data Documentation	46
3.5.4.1	_TAG	46
3.5.4.2	FALSE_VALUE	46
3.5.4.3	mValue	46
3.5.4.4	TRUE_VALUE	46
3.6	Asn1CharRange Class Reference	47
3.6.1	Detailed Description	47
3.6.2	Constructor & Destructor Documentation	47
3.6.2.1	Asn1CharRange	47
3.6.3	Member Function Documentation	48
3.6.3.1	GetCharAtIndex	48
3.6.3.2	GetCharIndex	48
3.6.3.3	validate	48
3.6.4	Member Data Documentation	49
3.6.4.1	mLower	49
3.6.4.2	mUpper	49
3.6.5	Property Documentation	49
3.6.5.1	MaxValue	49
3.7	Asn1CharSet Class Reference	50
3.7.1	Detailed Description	50
3.7.2	Constructor & Destructor Documentation	50
3.7.2.1	Asn1CharSet	50
3.7.3	Member Function Documentation	50
3.7.3.1	GetCharAtIndex	50
3.7.3.2	GetCharIndex	51
3.7.3.3	GetNumBitsPerChar	51
3.7.3.4	validate	51
3.7.4	Member Data Documentation	52
3.7.4.1	mABitsPerChar	52
3.7.4.2	mUBitsPerChar	52
3.7.5	Property Documentation	52
3.7.5.1	MaxValue	52
3.8	Asn1CharString Class Reference	53

3.8.1	Detailed Description	54
3.8.2	Constructor & Destructor Documentation	54
3.8.2.1	Asn1CharString	54
3.8.2.2	Asn1CharString	54
3.8.3	Member Function Documentation	54
3.8.3.1	Decode	54
3.8.3.2	Decode	54
3.8.3.3	Decode	55
3.8.3.4	Decode	55
3.8.3.5	DecodeByteToChar	55
3.8.3.6	DecodeXER	55
3.8.3.7	DecodeXML	55
3.8.3.8	Encode	56
3.8.3.9	Encode	56
3.8.3.10	Encode	56
3.8.3.11	Encode	56
3.8.3.12	Encode	57
3.8.3.13	Encode	57
3.8.3.14	Encode	57
3.8.3.15	Equals	58
3.8.3.16	Equals	58
3.8.3.17	GetHashCode	58
3.8.3.18	ToString	58
3.8.3.19	validate	58
3.8.4	Member Data Documentation	59
3.8.4.1	mStringBuffer	59
3.8.4.2	mValue	59
3.8.5	Property Documentation	59
3.8.5.1	Length	59
3.9	Asn1Choice Class Reference	60
3.9.1	Detailed Description	60
3.9.2	Constructor & Destructor Documentation	60
3.9.2.1	Asn1Choice	60
3.9.3	Member Function Documentation	60
3.9.3.1	Equals	60
3.9.3.2	GetElement	61
3.9.3.3	GetHashCode	61

3.9.3.4	SetElement	61
3.9.4	Member Data Documentation	61
3.9.4.1	choiceID	61
3.9.4.2	element	61
3.9.5	Property Documentation	61
3.9.5.1	ChoiceID	61
3.9.5.2	ElemName	62
3.10	Asn1ChoiceExt Class Reference	63
3.10.1	Detailed Description	63
3.10.2	Member Function Documentation	63
3.10.2.1	Decode	63
3.10.2.2	Decode	63
3.10.2.3	Encode	64
3.10.2.4	Encode	64
3.10.2.5	Encode	64
3.10.2.6	Encode	64
3.10.2.7	Encode	64
3.10.3	Member Data Documentation	65
3.10.3.1	choiceIndex	65
3.10.3.2	tag	65
3.11	Asn1ConsVioException Class Reference	66
3.11.1	Detailed Description	66
3.11.2	Constructor & Destructor Documentation	66
3.11.2.1	Asn1ConsVioException	66
3.11.2.2	Asn1ConsVioException	66
3.11.2.3	Asn1ConsVioException	66
3.12	Asn1DecodeBuffer Class Reference	67
3.12.1	Detailed Description	67
3.12.2	Member Function Documentation	68
3.12.2.1	AddCaptureBuffer	68
3.12.2.2	Capture	68
3.12.2.3	DecodeIntValue	68
3.12.2.4	DecodeOIDContents	68
3.12.2.5	DecodeRelOIDContents	68
3.12.2.6	GetInputStream	69
3.12.2.7	HexDump	69
3.12.2.8	Init	69

3.12.2.9	Mark	69
3.12.2.10	Read	69
3.12.2.11	Read	69
3.12.2.12	Read	70
3.12.2.13	Read2Bytes	70
3.12.2.14	Read4Bytes	70
3.12.2.15	ReadByte	71
3.12.2.16	RemoveCaptureBuffer	71
3.12.2.17	Reset	71
3.12.2.18	SetInputStream	71
3.12.2.19	Skip	71
3.12.3	Member Data Documentation	71
3.12.3.1	mByteCount	71
3.12.4	Property Documentation	72
3.12.4.1	ByteCount	72
3.12.4.2	LazyOpenTypeDecode	72
3.13	Asn1DiscreteCharSet Class Reference	73
3.13.1	Detailed Description	73
3.13.2	Constructor & Destructor Documentation	73
3.13.2.1	Asn1DiscreteCharSet	73
3.13.2.2	Asn1DiscreteCharSet	73
3.13.3	Member Function Documentation	73
3.13.3.1	GetCharAtIndex	73
3.13.3.2	GetCharIndex	74
3.13.3.3	helpValidate	74
3.13.3.4	validate	74
3.13.4	Property Documentation	75
3.13.4.1	MaxValue	75
3.14	Asn1EncodeBuffer Class Reference	76
3.14.1	Detailed Description	76
3.14.2	Member Function Documentation	77
3.14.2.1	BinDump	77
3.14.2.2	BinDump	77
3.14.2.3	Copy	77
3.14.2.4	Copy	77
3.14.2.5	Copy	77
3.14.2.6	EncodeIntSigned	78

3.14.2.7	EncodeIntUnsigned	78
3.14.2.8	GetInputStream	78
3.14.2.9	GetMinimalOctetsSigned	78
3.14.2.10	GetMinimalOctetsUnsigned	79
3.14.2.11	GetOutputStream	79
3.14.2.12	HexDump	79
3.14.2.13	HexDump	79
3.14.2.14	Reset	79
3.14.2.15	Write	79
3.14.3	Member Data Documentation	79
3.14.3.1	mSizeIncrement	79
3.14.3.2	SIZE_INCREMENT	80
3.14.4	Property Documentation	80
3.14.4.1	ByteArrayInputStream	80
3.14.4.2	MsgCopy	80
3.14.4.3	MsgLength	80
3.15	Asn1EndOfBufferException Class Reference	81
3.15.1	Detailed Description	81
3.15.2	Constructor & Destructor Documentation	81
3.15.2.1	Asn1EndOfBufferException	81
3.16	Asn1Enumerated Class Reference	82
3.16.1	Detailed Description	82
3.16.2	Constructor & Destructor Documentation	83
3.16.2.1	Asn1Enumerated	83
3.16.2.2	Asn1Enumerated	83
3.16.3	Member Function Documentation	83
3.16.3.1	Encode	83
3.16.3.2	Encode	83
3.16.3.3	Encode	84
3.16.3.4	Encode	84
3.16.3.5	Encode	84
3.16.3.6	Encode	84
3.16.3.7	Encode	84
3.16.3.8	Encode	85
3.16.3.9	Encode	85
3.16.3.10	Encode	85
3.16.3.11	Equals	85

3.16.3.12	Equals	85
3.16.3.13	GetHashCode	86
3.16.3.14	ParseValue	86
3.16.3.15	ToString	86
3.16.4	Member Data Documentation	86
3.16.4.1	_TAG	86
3.16.4.2	mValue	86
3.16.4.3	UNDEFINED	86
3.16.5	Property Documentation	87
3.16.5.1	Value	87
3.17	Asn1Exception Class Reference	88
3.17.1	Detailed Description	88
3.17.2	Constructor & Destructor Documentation	88
3.17.2.1	Asn1Exception	88
3.17.2.2	Asn1Exception	88
3.17.2.3	Asn1Exception	88
3.18	Asn1GeneralizedTime Class Reference	89
3.18.1	Detailed Description	89
3.18.2	Constructor & Destructor Documentation	89
3.18.2.1	Asn1GeneralizedTime	89
3.18.2.2	Asn1GeneralizedTime	89
3.18.2.3	Asn1GeneralizedTime	90
3.18.2.4	Asn1GeneralizedTime	90
3.18.3	Member Function Documentation	90
3.18.3.1	CompareTo	90
3.18.3.2	CompileString	90
3.18.3.3	Decode	90
3.18.3.4	Encode	91
3.18.3.5	Encode	91
3.18.3.6	ParseString	91
3.18.4	Member Data Documentation	91
3.18.4.1	_TAG	91
3.18.5	Property Documentation	91
3.18.5.1	Century	91
3.19	Asn1GeneralString Class Reference	93
3.19.1	Detailed Description	93
3.19.2	Constructor & Destructor Documentation	93

3.19.2.1	Asn1GeneralString	93
3.19.2.2	Asn1GeneralString	93
3.19.3	Member Function Documentation	93
3.19.3.1	Decode	93
3.19.3.2	Encode	94
3.19.3.3	Encode	94
3.19.4	Member Data Documentation	94
3.19.4.1	_TAG	94
3.20	Asn1GraphicString Class Reference	95
3.20.1	Detailed Description	95
3.20.2	Constructor & Destructor Documentation	95
3.20.2.1	Asn1GraphicString	95
3.20.2.2	Asn1GraphicString	95
3.20.3	Member Function Documentation	95
3.20.3.1	Decode	95
3.20.3.2	Encode	96
3.20.3.3	Encode	96
3.20.4	Member Data Documentation	96
3.20.4.1	_TAG	96
3.21	Asn1IA5String Class Reference	97
3.21.1	Detailed Description	97
3.21.2	Constructor & Destructor Documentation	97
3.21.2.1	Asn1IA5String	97
3.21.2.2	Asn1IA5String	97
3.21.3	Member Function Documentation	97
3.21.3.1	Decode	97
3.21.3.2	Encode	98
3.21.3.3	Encode	98
3.21.4	Member Data Documentation	98
3.21.4.1	_TAG	98
3.22	Asn1InputStream Interface Reference	99
3.22.1	Detailed Description	99
3.22.2	Member Function Documentation	99
3.22.2.1	Available	99
3.22.2.2	Close	99
3.22.2.3	Mark	99
3.22.2.4	MarkSupported	99

3.22.2.5	Reset	100
3.22.2.6	Skip	100
3.23	AsnInteger Class Reference	101
3.23.1	Detailed Description	102
3.23.2	Constructor & Destructor Documentation	102
3.23.2.1	AsnInteger	102
3.23.2.2	AsnInteger	102
3.23.3	Member Function Documentation	102
3.23.3.1	Decode	102
3.23.3.2	Decode	103
3.23.3.3	Decode	103
3.23.3.4	Decode	103
3.23.3.5	Decode	103
3.23.3.6	Decode	103
3.23.3.7	Decode	104
3.23.3.8	Decode	104
3.23.3.9	Decode16Bit	104
3.23.3.10	Decode32Bit	104
3.23.3.11	Decode8Bit	104
3.23.3.12	DecodeSigned	104
3.23.3.13	DecodeSigned	104
3.23.3.14	DecodeUnsigned	105
3.23.3.15	DecodeUnsigned	105
3.23.3.16	DecodeValue	105
3.23.3.17	DecodeValue	105
3.23.3.18	DecodeXER	105
3.23.3.19	DecodeXML	105
3.23.3.20	Encode	106
3.23.3.21	Encode	106
3.23.3.22	Encode	106
3.23.3.23	Encode	106
3.23.3.24	Encode	107
3.23.3.25	Encode	107
3.23.3.26	Encode	107
3.23.3.27	Encode	107
3.23.3.28	Encode	108
3.23.3.29	Encode	108

3.23.3.30	Encode	108
3.23.3.31	Encode	108
3.23.3.32	Encode	108
3.23.3.33	Encode	109
3.23.3.34	Encode	109
3.23.3.35	Encode	109
3.23.3.36	Encode16Bit	109
3.23.3.37	Encode32Bit	109
3.23.3.38	Encode8Bit	109
3.23.3.39	EncodeAttribute	110
3.23.3.40	EncodeSigned	110
3.23.3.41	EncodeSigned	110
3.23.3.42	EncodeUnsigned	110
3.23.3.43	EncodeUnsigned	110
3.23.3.44	EncodeValue	110
3.23.3.45	Equals	111
3.23.3.46	Equals	111
3.23.3.47	GetBitCount	111
3.23.3.48	GetBitCount	111
3.23.3.49	GetHashCode	111
3.23.3.50	GetUnsignedBitCount	112
3.23.3.51	GetUnsignedBitCount	112
3.23.3.52	ToString	112
3.23.4	Member Data Documentation	112
3.23.4.1	_TAG	112
3.23.4.2	mValue	112
3.24	Asn1InvalidArgException Class Reference	113
3.24.1	Detailed Description	113
3.24.2	Constructor & Destructor Documentation	113
3.24.2.1	Asn1InvalidArgException	113
3.25	Asn1InvalidChoiceOptionException Class Reference	114
3.25.1	Detailed Description	114
3.25.2	Constructor & Destructor Documentation	114
3.25.2.1	Asn1InvalidChoiceOptionException	114
3.25.2.2	Asn1InvalidChoiceOptionException	114
3.25.2.3	Asn1InvalidChoiceOptionException	114
3.26	Asn1InvalidEnumException Class Reference	115

3.26.1	Detailed Description	115
3.26.2	Constructor & Destructor Documentation	115
3.26.2.1	Asn1InvalidEnumException	115
3.26.2.2	Asn1InvalidEnumException	115
3.27	Asn1InvalidLengthException Class Reference	116
3.27.1	Detailed Description	116
3.27.2	Constructor & Destructor Documentation	116
3.27.2.1	Asn1InvalidLengthException	116
3.28	Asn1InvalidObjectIDException Class Reference	117
3.28.1	Detailed Description	117
3.28.2	Constructor & Destructor Documentation	117
3.28.2.1	Asn1InvalidObjectIDException	117
3.29	Asn1MessageBuffer Class Reference	118
3.29.1	Detailed Description	118
3.29.2	Member Function Documentation	118
3.29.2.1	AddNamedEventHandler	118
3.29.2.2	GetInputStream	118
3.29.2.3	InvokeCharacters	118
3.29.2.4	InvokeEndElement	119
3.29.2.5	InvokeStartElement	119
3.29.3	Property Documentation	119
3.29.3.1	EventHandlerList	119
3.30	Asn1MissingRequiredException Class Reference	120
3.30.1	Detailed Description	120
3.30.2	Constructor & Destructor Documentation	120
3.30.2.1	Asn1MissingRequiredException	120
3.30.2.2	Asn1MissingRequiredException	120
3.31	Asn1NamedEventHandler Interface Reference	121
3.31.1	Detailed Description	121
3.31.2	Member Function Documentation	121
3.31.2.1	Characters	121
3.31.2.2	EndElement	121
3.31.2.3	StartElement	122
3.32	Asn1Null Class Reference	123
3.32.1	Detailed Description	123
3.32.2	Member Function Documentation	123
3.32.2.1	Decode	123

3.32.2.2	Decode	123
3.32.2.3	Decode	124
3.32.2.4	Decode	124
3.32.2.5	DecodeXER	124
3.32.2.6	DecodeXML	124
3.32.2.7	Encode	124
3.32.2.8	Encode	125
3.32.2.9	Encode	125
3.32.2.10	Encode	125
3.32.2.11	Encode	125
3.32.2.12	Encode	126
3.32.2.13	Encode	126
3.32.2.14	Encode	126
3.32.2.15	Equals	126
3.32.2.16	ToString	126
3.32.3	Member Data Documentation	126
3.32.3.1	_TAG	126
3.32.3.2	NULL_VALUE	127
3.33	Asn1NumericString Class Reference	128
3.33.1	Detailed Description	128
3.33.2	Constructor & Destructor Documentation	128
3.33.2.1	Asn1NumericString	128
3.33.2.2	Asn1NumericString	128
3.33.3	Member Function Documentation	128
3.33.3.1	Decode	128
3.33.3.2	Decode	129
3.33.3.3	Decode	129
3.33.3.4	Encode	129
3.33.3.5	Encode	130
3.33.3.6	Encode	130
3.33.3.7	Encode	130
3.33.4	Member Data Documentation	130
3.33.4.1	_TAG	130
3.34	Asn1ObjectDescriptor Class Reference	131
3.34.1	Detailed Description	131
3.34.2	Constructor & Destructor Documentation	131
3.34.2.1	Asn1ObjectDescriptor	131

3.34.2.2	Asn1ObjectDescriptor	131
3.34.3	Member Function Documentation	131
3.34.3.1	Decode	131
3.34.3.2	Encode	132
3.34.3.3	Encode	132
3.34.4	Member Data Documentation	132
3.34.4.1	_TAG	132
3.35	Asn1ObjectIdentifier Class Reference	133
3.35.1	Detailed Description	133
3.35.2	Constructor & Destructor Documentation	134
3.35.2.1	Asn1ObjectIdentifier	134
3.35.2.2	Asn1ObjectIdentifier	134
3.35.3	Member Function Documentation	134
3.35.3.1	Append	134
3.35.3.2	Decode	134
3.35.3.3	Decode	134
3.35.3.4	Decode	134
3.35.3.5	Decode	135
3.35.3.6	DecodeXER	135
3.35.3.7	DecodeXML	135
3.35.3.8	Encode	135
3.35.3.9	Encode	136
3.35.3.10	Encode	136
3.35.3.11	Encode	136
3.35.3.12	Encode	136
3.35.3.13	Encode	137
3.35.3.14	Encode	137
3.35.3.15	Encode	137
3.35.3.16	Equals	137
3.35.3.17	GetHashCode	138
3.35.3.18	ToString	138
3.35.3.19	ToXMLValue	138
3.35.3.20	Validate	138
3.35.4	Member Data Documentation	138
3.35.4.1	_TAG	138
3.35.4.2	MAXSUBIDS	138
3.35.4.3	mValue	139

3.36 Asn1OctetString Class Reference	140
3.36.1 Detailed Description	141
3.36.2 Constructor & Destructor Documentation	141
3.36.2.1 Asn1OctetString	141
3.36.2.2 Asn1OctetString	141
3.36.2.3 Asn1OctetString	141
3.36.2.4 Asn1OctetString	141
3.36.3 Member Function Documentation	142
3.36.3.1 CompareTo	142
3.36.3.2 Decode	142
3.36.3.3 Decode	142
3.36.3.4 Decode	142
3.36.3.5 Decode	142
3.36.3.6 Decode	143
3.36.3.7 Decode	143
3.36.3.8 Decode	143
3.36.3.9 DecodeContent	143
3.36.3.10 DecodeXER	144
3.36.3.11 DecodeXML	144
3.36.3.12 Encode	144
3.36.3.13 Encode	144
3.36.3.14 Encode	145
3.36.3.15 Encode	145
3.36.3.16 Encode	145
3.36.3.17 Encode	145
3.36.3.18 Encode	146
3.36.3.19 Encode	146
3.36.3.20 Encode	146
3.36.3.21 Encode	146
3.36.3.22 Encode	146
3.36.3.23 Encode	147
3.36.3.24 Encode	147
3.36.3.25 EncodeAttribute	147
3.36.3.26 EncodeBase64Binary	147
3.36.3.27 EncodeContent	148
3.36.3.28 Equals	148
3.36.3.29 Equals	148

3.36.3.30	Equals	148
3.36.3.31	GetHashCode	149
3.36.3.32	GetMderLength	149
3.36.3.33	toInputStream	149
3.36.3.34	ToString	149
3.36.4	Member Data Documentation	149
3.36.4.1	_TAG	149
3.36.4.2	mValue	149
3.36.5	Property Documentation	149
3.36.5.1	Length	149
3.37	Asn1OpenExt Class Reference	151
3.37.1	Detailed Description	151
3.37.2	Member Function Documentation	151
3.37.2.1	Decode	151
3.37.2.2	Decode	152
3.37.2.3	DecodeComponent	152
3.37.2.4	DecodeEventComponent	152
3.37.2.5	DecodeExtension	152
3.37.2.6	DecodeOpenType	153
3.37.2.7	DecodeOpenType	153
3.37.2.8	Encode	153
3.37.2.9	Encode	153
3.37.2.10	Encode	154
3.37.2.11	Encode	154
3.37.2.12	Encode	154
3.37.2.13	Encode	154
3.37.2.14	Encode	154
3.37.2.15	Encode	155
3.37.2.16	EncodeExtBits	155
3.37.2.17	EncodeExtBits	155
3.37.2.18	HasPresentExtensions	155
3.37.2.19	SetOpenType	155
3.37.2.20	ShrinkArray	155
3.37.2.21	ToString	156
3.37.3	Member Data Documentation	156
3.37.3.1	mValue	156
3.37.4	Property Documentation	156

3.37.4.1	AsnTypeName	156
3.38	Asn1OpenType Class Reference	157
3.38.1	Detailed Description	158
3.38.2	Constructor & Destructor Documentation	158
3.38.2.1	Asn1OpenType	158
3.38.2.2	Asn1OpenType	158
3.38.2.3	Asn1OpenType	159
3.38.2.4	Asn1OpenType	159
3.38.2.5	Asn1OpenType	159
3.38.2.6	Asn1OpenType	159
3.38.2.7	Asn1OpenType	159
3.38.2.8	Asn1OpenType	160
3.38.3	Member Function Documentation	160
3.38.3.1	Decode	160
3.38.3.2	Decode	160
3.38.3.3	Decode	160
3.38.3.4	Decode	160
3.38.3.5	DecodeExtension	161
3.38.3.6	Encode	161
3.38.3.7	Encode	161
3.38.3.8	Encode	161
3.38.3.9	Encode	162
3.38.3.10	Encode	162
3.38.3.11	Encode	162
3.38.3.12	Encode	162
3.38.3.13	Encode	163
3.38.3.14	Encode	163
3.38.3.15	Encode	163
3.38.3.16	EncodeAsExtension	163
3.38.3.17	EncodeAsExtension	164
3.38.3.18	EncodeAsExtension	164
3.38.3.19	GetCharData	164
3.38.3.20	GetDataEncoding	164
3.38.3.21	GetSaxHandler	164
3.38.3.22	GetSaxHandler	165
3.38.3.23	SetBinaryData	165
3.38.3.24	SetCharData	165

3.38.3.25	SetCharData	165
3.38.3.26	ToString	165
3.38.4	Member Data Documentation	166
3.38.4.1	dataEncoding	166
3.38.4.2	mEncodeBuffer	166
3.38.4.3	mLength	166
3.38.5	Property Documentation	166
3.38.5.1	AsnTypeName	166
3.39	Asn1OutputStream Class Reference	167
3.39.1	Detailed Description	167
3.39.2	Constructor & Destructor Documentation	167
3.39.2.1	Asn1OutputStream	167
3.39.3	Member Function Documentation	168
3.39.3.1	Close	168
3.39.3.2	Flush	168
3.39.3.3	Read	168
3.39.3.4	Seek	168
3.39.3.5	SetLength	169
3.39.3.6	Write	169
3.39.3.7	Write	170
3.39.3.8	Write2Bytes	170
3.39.3.9	Write4Bytes	170
3.39.3.10	WriteByte	170
3.39.3.11	WriteByte	171
3.39.4	Member Data Documentation	171
3.39.4.1	os	171
3.39.5	Property Documentation	171
3.39.5.1	CanRead	171
3.39.5.2	CanSeek	171
3.39.5.3	CanWrite	171
3.39.5.4	Context	171
3.39.5.5	Length	171
3.39.5.6	Position	172
3.40	Asn1PrintableString Class Reference	173
3.40.1	Detailed Description	173
3.40.2	Constructor & Destructor Documentation	173
3.40.2.1	Asn1PrintableString	173

3.40.2.2	Asn1PrintableString	173
3.40.3	Member Function Documentation	173
3.40.3.1	Decode	173
3.40.3.2	Encode	174
3.40.3.3	Encode	174
3.40.4	Member Data Documentation	174
3.40.4.1	_TAG	174
3.41	Asn1Real Class Reference	175
3.41.1	Detailed Description	176
3.41.2	Constructor & Destructor Documentation	176
3.41.2.1	Asn1Real	176
3.41.2.2	Asn1Real	176
3.41.3	Member Function Documentation	176
3.41.3.1	Decode	176
3.41.3.2	Decode	176
3.41.3.3	Decode	176
3.41.3.4	Decode	177
3.41.3.5	Decode	177
3.41.3.6	DecodeDouble	177
3.41.3.7	DecodeSingle	177
3.41.3.8	DecodeXER	178
3.41.3.9	DecodeXML	178
3.41.3.10	Encode	178
3.41.3.11	Encode	178
3.41.3.12	Encode	179
3.41.3.13	Encode	179
3.41.3.14	Encode	179
3.41.3.15	Encode	179
3.41.3.16	Encode	180
3.41.3.17	Encode	180
3.41.3.18	Encode	180
3.41.3.19	EncodeAttribute	180
3.41.3.20	EncodeDouble	180
3.41.3.21	EncodeSingle	181
3.41.3.22	EncodeValue	181
3.41.3.23	Equals	181
3.41.3.24	Equals	181

3.41.3.25	GetHashCode	181
3.41.3.26	NormalizedRealValueToString	182
3.41.3.27	ToString	182
3.41.4	Member Data Documentation	182
3.41.4.1	_TAG	182
3.41.4.2	mValue	182
3.42	Asn1Real10 Class Reference	183
3.42.1	Detailed Description	183
3.42.2	Constructor & Destructor Documentation	183
3.42.2.1	Asn1Real10	183
3.42.2.2	Asn1Real10	183
3.42.3	Member Function Documentation	184
3.42.3.1	ConvertToDecimal	184
3.42.3.2	ConvertToNR3Form	184
3.42.3.3	Decode	184
3.42.3.4	Decode	184
3.42.3.5	Decode	184
3.42.3.6	Encode	185
3.42.3.7	Encode	185
3.42.3.8	Encode	185
3.42.3.9	Encode	185
3.42.3.10	Encode	186
3.42.3.11	Encode	186
3.42.3.12	Encode	186
3.42.3.13	Encode	186
3.42.3.14	Encode	187
3.42.3.15	EncodeAttribute	187
3.42.3.16	GetNumberForm	187
3.42.3.17	GetNumberForm	188
3.42.4	Member Data Documentation	188
3.42.4.1	_TAG	188
3.43	Asn1RelativeOID Class Reference	189
3.43.1	Detailed Description	189
3.43.2	Constructor & Destructor Documentation	189
3.43.2.1	Asn1RelativeOID	189
3.43.2.2	Asn1RelativeOID	189
3.43.3	Member Function Documentation	190

3.43.3.1	Decode	190
3.43.3.2	Decode	190
3.43.3.3	Decode	190
3.43.3.4	DecodeXER	190
3.43.3.5	DecodeXML	190
3.43.3.6	Encode	191
3.43.3.7	Encode	191
3.43.3.8	Encode	191
3.43.3.9	Encode	191
3.43.3.10	Encode	192
3.43.3.11	Encode	192
3.43.3.12	Encode	192
3.43.3.13	Validate	192
3.43.4	Member Data Documentation	193
3.43.4.1	_TAG	193
3.44	Asn1SeqOrderException Class Reference	194
3.44.1	Detailed Description	194
3.44.2	Constructor & Destructor Documentation	194
3.44.2.1	Asn1SeqOrderException	194
3.45	Asn1Status Class Reference	195
3.45.1	Detailed Description	195
3.45.2	Member Data Documentation	195
3.45.2.1	INDEFLEN	195
3.46	Asn1T61String Class Reference	196
3.46.1	Detailed Description	196
3.46.2	Constructor & Destructor Documentation	196
3.46.2.1	Asn1T61String	196
3.46.2.2	Asn1T61String	196
3.46.3	Member Function Documentation	196
3.46.3.1	Decode	196
3.46.3.2	Encode	197
3.46.3.3	Encode	197
3.46.4	Member Data Documentation	197
3.46.4.1	_TAG	197
3.47	Asn1Tag Class Reference	198
3.47.1	Detailed Description	198
3.47.2	Constructor & Destructor Documentation	199

3.47.2.1	Asn1Tag	199
3.47.2.2	Asn1Tag	199
3.47.3	Member Function Documentation	199
3.47.3.1	Equals	199
3.47.3.2	Equals	199
3.47.3.3	IsEOC	199
3.47.3.4	ToString	200
3.47.4	Member Data Documentation	200
3.47.4.1	APPL	200
3.47.4.2	Bit8Mask	200
3.47.4.3	ClassMask	200
3.47.4.4	CONS	200
3.47.4.5	CTXT	200
3.47.4.6	ENUM	200
3.47.4.7	EOC	200
3.47.4.8	EXPL	200
3.47.4.9	EXTIDCODE	200
3.47.4.10	FormMask	201
3.47.4.11	IDMask	201
3.47.4.12	IMPL	201
3.47.4.13	L7BitsMask	201
3.47.4.14	mClass	201
3.47.4.15	mForm	201
3.47.4.16	mIDCode	201
3.47.4.17	PRIM	201
3.47.4.18	PRIV	201
3.47.4.19	SEQUENCE	201
3.47.4.20	SET	201
3.47.4.21	UNIV	201
3.47.5	Property Documentation	202
3.47.5.1	Constructed	202
3.48	Asn1Time Class Reference	203
3.48.1	Detailed Description	203
3.48.2	Constructor & Destructor Documentation	203
3.48.2.1	Asn1Time	203
3.48.3	Member Function Documentation	203
3.48.3.1	Decode	203

3.48.3.2	Encode	203
3.48.3.3	Encode	204
3.48.4	Member Data Documentation	204
3.48.4.1	January	204
3.49	Asn1TraceHandler Class Reference	205
3.49.1	Detailed Description	205
3.49.2	Constructor & Destructor Documentation	205
3.49.2.1	Asn1TraceHandler	205
3.49.2.2	Asn1TraceHandler	205
3.49.3	Member Function Documentation	205
3.49.3.1	Characters	205
3.49.3.2	EndElement	206
3.49.3.3	StartElement	206
3.50	Asn1Type Class Reference	207
3.50.1	Detailed Description	209
3.50.2	Member Function Documentation	209
3.50.2.1	_SetKey	209
3.50.2.2	_SetKey2	209
3.50.2.3	Decode	209
3.50.2.4	Decode	209
3.50.2.5	Decode	210
3.50.2.6	Decode	210
3.50.2.7	Decode	210
3.50.2.8	Decode	211
3.50.2.9	Decode	211
3.50.2.10	Decode	211
3.50.2.11	Decode	211
3.50.2.12	Decode	211
3.50.2.13	DecodeXML	212
3.50.2.14	Encode	212
3.50.2.15	Encode	212
3.50.2.16	Encode	213
3.50.2.17	Encode	213
3.50.2.18	Encode	213
3.50.2.19	Encode	214
3.50.2.20	Encode	214
3.50.2.21	Encode	215

3.50.2.22	Encode	215
3.50.2.23	Encode	215
3.50.2.24	Encode	216
3.50.2.25	Encode	216
3.50.2.26	Encode	216
3.50.2.27	EncodeAsOpenType	217
3.50.2.28	EncodeAttribute	217
3.50.2.29	Equals	217
3.50.2.30	GetNonParameterizedTypeName	217
3.50.2.31	GetTypeName	217
3.50.2.32	Indent	217
3.50.2.33	IsOpenType	218
3.50.2.34	MatchTag	218
3.50.2.35	MatchTag	218
3.50.2.36	MatchTypeName	219
3.50.2.37	Pdiag	219
3.50.2.38	Print	219
3.50.2.39	Print	219
3.50.2.40	SetNonParameterizedTypeName	219
3.50.2.41	SetOpenType	219
3.50.3	Member Data Documentation	220
3.50.3.1	_TAG	220
3.50.3.2	BIT_STRING	220
3.50.3.3	BMPString	220
3.50.3.4	BOOLEAN	220
3.50.3.5	DATE	220
3.50.3.6	ENUMERATED	220
3.50.3.7	EOC	220
3.50.3.8	EXTERNAL	220
3.50.3.9	GeneralString	220
3.50.3.10	GeneralTime	221
3.50.3.11	GraphicString	221
3.50.3.12	IA5String	221
3.50.3.13	INTEGER	221
3.50.3.14	NULL	221
3.50.3.15	NumericString	221
3.50.3.16	OBJECT_IDENTIFIER	221

3.50.3.17	ObjectDescriptor	221
3.50.3.18	OCTET_STRING	221
3.50.3.19	OpenType	221
3.50.3.20	PrintableString	221
3.50.3.21	REAL	221
3.50.3.22	RELATIVE_OID_IRI	222
3.50.3.23	RelativeOID	222
3.50.3.24	SEQUENCE	222
3.50.3.25	SET	222
3.50.3.26	T61String	222
3.50.3.27	TeletexString	222
3.50.3.28	TIME	222
3.50.3.29	UniversalString	222
3.50.3.30	UTCTime	222
3.50.3.31	UTF8String	222
3.50.3.32	VideotexString	222
3.50.3.33	VisibleString	222
3.50.4	Property Documentation	223
3.50.4.1	AsnTypeName	223
3.50.4.2	Length	223
3.51	Asn1TypeIF Interface Reference	224
3.51.1	Detailed Description	224
3.51.2	Member Function Documentation	224
3.51.2.1	Decode	224
3.51.2.2	Decode	224
3.51.2.3	Decode	225
3.51.2.4	Decode	225
3.51.2.5	Decode	225
3.51.2.6	Encode	225
3.51.2.7	Encode	226
3.51.2.8	Encode	226
3.51.2.9	Encode	226
3.51.2.10	Encode	227
3.51.2.11	Encode	227
3.51.2.12	Encode	227
3.51.2.13	Encode	228
3.51.2.14	Encode	228

3.51.2.15	IsOpenType	228
3.51.2.16	Print	228
3.51.2.17	SetOpenType	229
3.52	Asn1UniversalString Class Reference	230
3.52.1	Detailed Description	231
3.52.2	Constructor & Destructor Documentation	231
3.52.2.1	Asn1UniversalString	231
3.52.2.2	Asn1UniversalString	231
3.52.2.3	Asn1UniversalString	231
3.52.3	Member Function Documentation	232
3.52.3.1	Decode	232
3.52.3.2	Decode	232
3.52.3.3	Decode	232
3.52.3.4	Decode	232
3.52.3.5	Decode	232
3.52.3.6	Decode	233
3.52.3.7	Decode	233
3.52.3.8	Decode	233
3.52.3.9	Decode	234
3.52.3.10	Decode	234
3.52.3.11	DecodeXER	234
3.52.3.12	DecodeXML	234
3.52.3.13	Encode	234
3.52.3.14	Encode	235
3.52.3.15	Encode	235
3.52.3.16	Encode	236
3.52.3.17	Encode	236
3.52.3.18	Encode	236
3.52.3.19	Encode	237
3.52.3.20	Encode	237
3.52.3.21	Encode	237
3.52.3.22	Encode	238
3.52.3.23	Encode	238
3.52.3.24	Encode	238
3.52.3.25	Encode	238
3.52.3.26	Encode	239
3.52.3.27	Encode	239

3.52.3.28	Encode	239
3.52.3.29	Encode	240
3.52.3.30	Encode	240
3.52.3.31	Encode	240
3.52.3.32	EncodeData	240
3.52.3.33	Equals	241
3.52.3.34	GetHashCode	241
3.52.3.35	ToString	241
3.52.3.36	validate	241
3.52.4	Member Data Documentation	241
3.52.4.1	_TAG	241
3.52.4.2	BITSPERCHAR	241
3.52.4.3	mStringBuffer	241
3.52.4.4	mValue	242
3.52.5	Property Documentation	242
3.52.5.1	Length	242
3.53	Asn1UTCTime Class Reference	243
3.53.1	Detailed Description	243
3.53.2	Constructor & Destructor Documentation	243
3.53.2.1	Asn1UTCTime	243
3.53.2.2	Asn1UTCTime	243
3.53.2.3	Asn1UTCTime	244
3.53.2.4	Asn1UTCTime	244
3.53.3	Member Function Documentation	244
3.53.3.1	Clear	244
3.53.3.2	CompareTo	244
3.53.3.3	CompileString	244
3.53.3.4	Decode	245
3.53.3.5	Encode	245
3.53.3.6	Encode	245
3.53.3.7	Init	245
3.53.3.8	ParseString	245
3.53.3.9	SetTime	246
3.53.4	Member Data Documentation	246
3.53.4.1	_TAG	246
3.53.5	Property Documentation	246
3.53.5.1	Fraction	246

3.53.5.2	Year	246
3.54	Asn1UTF8String Class Reference	247
3.54.1	Detailed Description	247
3.54.2	Constructor & Destructor Documentation	247
3.54.2.1	Asn1UTF8String	247
3.54.2.2	Asn1UTF8String	248
3.54.3	Member Function Documentation	248
3.54.3.1	Decode	248
3.54.3.2	Decode	248
3.54.3.3	Decode	248
3.54.3.4	Decode	248
3.54.3.5	Decode	249
3.54.3.6	Decode	249
3.54.3.7	DecodeUTF8	249
3.54.3.8	DecodeUTF8	249
3.54.3.9	Encode	250
3.54.3.10	Encode	250
3.54.3.11	Encode	250
3.54.3.12	Encode	251
3.54.3.13	Encode	251
3.54.3.14	Encode	251
3.54.3.15	Encode	251
3.54.3.16	Encode	252
3.54.3.17	Encode	252
3.54.3.18	Encode	252
3.54.3.19	Encode	252
3.54.3.20	SetAnyAttribute	253
3.54.4	Member Data Documentation	253
3.54.4.1	_TAG	253
3.55	Asn1Util Class Reference	254
3.55.1	Detailed Description	254
3.55.2	Member Function Documentation	254
3.55.2.1	BCDToString	254
3.55.2.2	DecodeBase64Array	255
3.55.2.3	EncodeBase64Array	255
3.55.2.4	GetAddressBytes	255
3.55.2.5	GetBytesCount	255

3.55.2.6	GetUlongBytesCount	256
3.55.2.7	IsLimited	256
3.55.2.8	StringToBCD	256
3.55.2.9	StringToTBCD	256
3.55.2.10	StripWhitespace	257
3.55.2.11	TBCDToString	257
3.55.2.12	ToArray	257
3.55.2.13	ToByteArray	257
3.55.2.14	ToCharArray	257
3.55.2.15	ToHexString	258
3.55.2.16	ToHexString	258
3.55.2.17	ToHexString	258
3.55.2.18	URShift	259
3.55.2.19	URShift	259
3.55.2.20	URShift	259
3.55.2.21	URShift	259
3.55.2.22	WriteStackTrace	260
3.56	Asn1Value Class Reference	261
3.56.1	Detailed Description	261
3.56.2	Member Function Documentation	261
3.56.2.1	ParseString	261
3.56.2.2	ParseString	261
3.56.2.3	StringEqualsBytes	262
3.56.2.4	StringEqualsBytes	262
3.57	Asn1ValueParseException Class Reference	263
3.57.1	Detailed Description	263
3.57.2	Constructor & Destructor Documentation	263
3.57.2.1	Asn1ValueParseException	263
3.57.2.2	Asn1ValueParseException	263
3.58	Asn1VarWidthCharString Class Reference	264
3.58.1	Detailed Description	264
3.58.2	Constructor & Destructor Documentation	264
3.58.2.1	Asn1VarWidthCharString	264
3.58.2.2	Asn1VarWidthCharString	264
3.58.3	Member Function Documentation	265
3.58.3.1	Decode	265
3.58.3.2	Decode	265

3.58.3.3	Decode	265
3.58.3.4	Encode	265
3.58.3.5	Encode	266
3.58.3.6	Encode	266
3.58.3.7	Encode	266
3.58.3.8	Encode	266
3.58.4	Member Data Documentation	267
3.58.4.1	BITSPERCHAR_A	267
3.58.4.2	BITSPERCHAR_U	267
3.59	Asn1VideotexString Class Reference	268
3.59.1	Detailed Description	268
3.59.2	Constructor & Destructor Documentation	268
3.59.2.1	Asn1VideotexString	268
3.59.2.2	Asn1VideotexString	268
3.59.3	Member Function Documentation	268
3.59.3.1	Decode	268
3.59.3.2	Encode	269
3.59.3.3	Encode	269
3.59.4	Member Data Documentation	269
3.59.4.1	_TAG	269
3.60	Asn1VisibleString Class Reference	270
3.60.1	Detailed Description	270
3.60.2	Constructor & Destructor Documentation	270
3.60.2.1	Asn1VisibleString	270
3.60.2.2	Asn1VisibleString	270
3.60.3	Member Function Documentation	270
3.60.3.1	Decode	270
3.60.3.2	Encode	271
3.60.3.3	Encode	271
3.60.4	Member Data Documentation	271
3.60.4.1	_TAG	271
3.61	BigInteger Class Reference	272
3.61.1	Detailed Description	272
3.61.2	Constructor & Destructor Documentation	272
3.61.2.1	BigInteger	272
3.61.2.2	BigInteger	272
3.61.2.3	BigInteger	273

3.61.2.4	BigInteger	273
3.61.2.5	BigInteger	273
3.61.3	Member Function Documentation	273
3.61.3.1	Add	273
3.61.3.2	BitLength	273
3.61.3.3	CompareTo	274
3.61.3.4	Equals	274
3.61.3.5	Equals	274
3.61.3.6	GetData	274
3.61.3.7	GetHashCode	274
3.61.3.8	Init	275
3.61.3.9	IsNegative	275
3.61.3.10	LongValue	275
3.61.3.11	operator BigInteger	275
3.61.3.12	SecureDelete	275
3.61.3.13	SetData	275
3.61.3.14	Subtract	276
3.61.3.15	ToString	276
3.61.3.16	ToString	276
3.62	BooleanHolder Class Reference	277
3.62.1	Detailed Description	277
3.62.2	Constructor & Destructor Documentation	277
3.62.2.1	BooleanHolder	277
3.62.2.2	BooleanHolder	277
3.62.3	Member Data Documentation	277
3.62.3.1	mValue	277
3.63	Diag Class Reference	278
3.63.1	Detailed Description	278
3.63.2	Member Function Documentation	278
3.63.2.1	HexDump	278
3.63.2.2	HexDump	279
3.63.2.3	HexDump	279
3.63.2.4	Instance	279
3.63.2.5	IsEnabled	279
3.63.2.6	IsEnabled	279
3.63.2.7	Println	279
3.63.2.8	Println	280

3.63.2.9	Prtln	280
3.63.2.10	Prtln	280
3.63.2.11	Prtln	280
3.63.2.12	Prtln	280
3.63.2.13	SetEnabled	281
3.63.2.14	SetTraceLevel	281
3.63.2.15	SetTraceLevel2	281
3.63.3	Property Documentation	281
3.63.3.1	PrintStream	281
3.64	IntHolder Class Reference	282
3.64.1	Detailed Description	282
3.64.2	Constructor & Destructor Documentation	282
3.64.2.1	IntHolder	282
3.64.2.2	IntHolder	282
3.64.3	Member Data Documentation	282
3.64.3.1	mValue	282
3.65	SaxHandler Class Reference	283
3.65.1	Detailed Description	283
3.65.2	Member Function Documentation	283
3.65.2.1	Characters	283
3.65.2.2	EndElement	283
3.65.2.3	StartElement	283
3.66	StringBufferExt Class Reference	284
3.66.1	Detailed Description	284
3.66.2	Member Function Documentation	284
3.66.2.1	Replace	284
3.67	Tokenizer Class Reference	285
3.67.1	Detailed Description	285
3.67.2	Constructor & Destructor Documentation	285
3.67.2.1	Tokenizer	285
3.67.2.2	Tokenizer	285
3.67.2.3	Tokenizer	285
3.67.3	Member Function Documentation	286
3.67.3.1	HasMoreTokens	286
3.67.3.2	MoveNext	286
3.67.3.3	NextToken	286
3.67.3.4	NextToken	286

3.67.3.5	RemainingString	286
3.67.3.6	Reset	286
3.67.4	Property Documentation	287
3.67.4.1	Count	287
3.67.4.2	Current	287

Chapter 1

ASN1C C# Runtime Library

The ASN.1 C# runtime library uses the `Com.Objsys.Asn1.Runtime` namespace. This namespace contains the implementation of the following rules:

- BER (As per ITU-T X.690 standard)
- CER (As per ITU-T X.690 standard)
- DER (As per ITU-T X.690 standard)
- MDER (As per ISO/IEEE 11073-2101:2004 standard)
- PER (As per ITU-T X.691 standard)
- XER (As per ITU-T X.693 standard)
- XML (As per asn2xsd converter)

Chapter 2

Namespace Documentation

2.1 Package Com.Objsys.Asn1.Runtime

Classes

- class [Asn18BitCharString](#)
- class [Asn1BigInteger](#)
- class [Asn1BitString](#)
- class [Asn1BMPString](#)
- class [Asn1Boolean](#)
- class [Asn1CharRange](#)
- class [Asn1CharSet](#)
- class [Asn1CharString](#)
- class [Asn1Choice](#)
- class [Asn1ChoiceExt](#)
- class [Asn1ConsVioException](#)
- class [Asn1DecodeBuffer](#)
- class [Asn1DiscreteCharSet](#)
- class [Asn1EncodeBuffer](#)
- class [Asn1EndOfBufferException](#)
- class [Asn1Enumerated](#)
- class [Asn1Exception](#)
- class [Asn1GeneralizedTime](#)
- class [Asn1GeneralString](#)
- class [Asn1GraphicString](#)
- class [Asn1IA5String](#)
- interface [Asn1InputStream](#)
- class [Asn1Integer](#)
- class [Asn1InvalidArgException](#)
- class [Asn1InvalidChoiceOptionException](#)
- class [Asn1InvalidEnumException](#)
- class [Asn1InvalidLengthException](#)
- class [Asn1InvalidObjectIDException](#)
- class [Asn1MessageBuffer](#)
- class [Asn1MissingRequiredException](#)
- interface [Asn1NamedEventHandler](#)

- class [Asn1Null](#)
- class [Asn1NumericString](#)
- class [Asn1ObjectDescriptor](#)
- class [Asn1ObjectIdentifier](#)
- class [Asn1OctetString](#)
- class [Asn1OpenExt](#)
- class [Asn1OpenType](#)
- class [Asn1OutputStream](#)
- class [Asn1PrintableString](#)
- class [Asn1Real](#)
- class [Asn1Real10](#)
- class [Asn1RelativeOID](#)
- class [Asn1SeqOrderException](#)
- class [Asn1Status](#)
- class [Asn1T61String](#)
- class [Asn1Tag](#)
- class [Asn1Time](#)

This class is used for all TIME types that are not one of the useful time types (viz.

- class [Asn1TraceHandler](#)
- class [Asn1Type](#)
- interface [Asn1TypeIF](#)
- class [Asn1UniversalString](#)
- class [Asn1UTCTime](#)
- class [Asn1UTF8String](#)
- class [Asn1Util](#)
- class [Asn1Value](#)
- class [Asn1ValueParseException](#)
- class [Asn1VarWidthCharString](#)
- class [Asn1VideotexString](#)
- class [Asn1VisibleString](#)
- class [BigInteger](#)
- class [BooleanHolder](#)
- class [Diag](#)
- class [IntHolder](#)
- class [StringBufferExt](#)
- class [Tokenizer](#)

2.1.1 Detailed Description

The ASN.1 C# runtime library uses the [Com.Objsys.Asn1.Runtime](#) namespace. This namespace contains the implementation of the following rules:

- BER (As per ITU-T X.690 standard)
- CER (As per ITU-T X.690 standard)
- DER (As per ITU-T X.690 standard)
- MDER (As per ISO/IEEE 11073-2101:2004 standard)
- PER (As per ITU-T X.691 standard)

- XER (As per ITU-T X.693 standard)
- XML (As per asn2xsd converter)

Chapter 3

Class Documentation

3.1 Asn18BitCharString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1CharString](#).

Inherited by [Asn1IA5String](#), [Asn1NumericString](#), [Asn1PrintableString](#), [Asn1Time](#), and [Asn1VisibleString](#).

Public Member Functions

- void [Decode](#) (Asn1OerDecodeBuffer buffer, int length)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- virtual void [Encode](#) (Asn1OerEncodeBuffer buffer, bool withLength)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerOutputStream outs)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)

Public Attributes

- const int [BITSPERCHAR_A](#) = 8
- const int [BITSPERCHAR_U](#) = 7

Protected Member Functions

- internal [Asn18BitCharString](#) (System.String data, short typeCode)
- internal [Asn18BitCharString](#) (short typeCode)

3.1.1 Detailed Description

This is an abstract base class for holding the ASN.1 8-bit character string types (IA5String, VisibleString, PrintableString, etc.).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 internal Asn18BitCharString (short *typeCode*) [protected]

The default constructor creates an empty string object.

Parameters

typeCode Universal ID code for ASN.1 character string

3.1.2.2 internal Asn18BitCharString (System.String *data*, short *typeCode*) [protected]

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data Character string

typeCode Universal ID code for ASN.1 character string

3.1.3 Member Function Documentation

3.1.3.1 void Decode (Asn1OerDecodeBuffer *buffer*, int *length*)

<summary> Decode the value in accordance with OER.

This class's implementation decodes a string of the given length. This method is not virtual as I don't see any reason for overriding it. </summary>

Parameters

length Length of string to decode

3.1.3.2 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode the value in accordance with OER.

This class's implementation decodes the string with a length determinant. If a subclass should be decoded without a length determinant, it should override this method to invoke `Decode(buffer, length)`. Subclasses may override this to add constraint checks after invoking this method to decode the string.

Reimplemented from [Asn1Type](#).

3.1.3.3 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*) [virtual]

This overloaded version of the `Decode` method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present and an optional permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Decode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

3.1.3.4 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Decode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)

3.1.3.5 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1NumericString](#).

3.1.3.6 virtual void Encode (Asn1OerEncodeBuffer *buffer*, bool *withLength*) [virtual]

Encode the string, with or without a length determinant.

Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters

- withLength* true if a length determinant should be encoded.

3.1.3.7 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

Encode the value in accordance with OER.

This class's implementation invokes `Encode(buffer, true)` to encode the string with a length determinant. If a subclass should be encoded without a length determinant, it should override this to invoke `Encode(buffer, false)`.

Reimplemented from [Asn1Type](#).

3.1.3.8 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet, long lower, long upper) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value directly into the stream, in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present and an optional permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Encode message stream object

charSet Object representing permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.1.3.9 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream. This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Encode message stream object

charSet Object representing permitted alphabet constraint character set (optional)

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.1.3.10 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream. This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Encode message stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.1.3.11 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present and an optional permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Encode message buffer object

charSet Object representing permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.1.3.12 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Encode message buffer object

charSet Object representing permitted alphabet constraint character set (optional)

3.1.3.13 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1NumericString](#).

3.1.4 Member Data Documentation

3.1.4.1 const int BITSPERCHAR_A = 8

The `BITSPERCHAR_A` constant specifies the number of bits per character for PER (aligned).

3.1.4.2 `const int BITSPERCHAR_U = 7`

The `BITSPERCHAR_U` constant specifies the number of bits per character for PER (unaligned).

3.2 Asn1BigInteger Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1BigInteger](#) (System.String value, int radix)
- [Asn1BigInteger](#) (System.String value)
- [Asn1BigInteger](#) ([BigInteger](#) value)
- [Asn1BigInteger](#) ()
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- void [Decode](#) (Asn1PerDecodeBuffer buffer, [BigInteger](#) lower, [BigInteger](#) upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [DecodeSigned](#) (Asn1OerDecodeBuffer buffer, int octets)
- void [DecodeSigned](#) (Asn1OerDecodeBuffer buffer)
- void [DecodeUnsigned](#) (Asn1OerDecodeBuffer buffer, int octets)
- void [DecodeUnsigned](#) (Asn1OerDecodeBuffer buffer)
- [BigInteger DecodeValue](#) ([Asn1DecodeBuffer](#) buffer, int length)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- void [Encode](#) (Asn1PerEncodeBuffer buffer, [BigInteger](#) lower, [BigInteger](#) upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, String attrName)
- void [EncodeSigned](#) (Asn1OerEncodeBuffer buffer)
- void [EncodeSigned](#) (Asn1OerEncodeBuffer buffer, int octets)
- void [EncodeUnsigned](#) (Asn1OerEncodeBuffer buffer, int octets)
- void [EncodeUnsigned](#) (Asn1OerEncodeBuffer buffer)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (long value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

Public Attributes

- [BigInteger mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag _TAG](#) = new [Asn1Tag](#)([Asn1Tag.UNIV](#), [Asn1Tag.PRIM](#), [Asn1Type.INTEGER](#))

Static Protected Member Functions

- static internal int `EncodeValue` (`Asn1EncodeBuffer` buffer, `BigInteger` ivalue, bool doCopy)

3.2.1 Detailed Description

This class represents an ASN.1 INTEGER built-in type. In this case, the values can be greater than 64 bits in size. This class is used in generated source code if the <bigInteger> qualifier is specified in a compiler configuration file.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 `Asn1BigInteger ()`

The default constructor sets the big integer value object reference to null.

3.2.2.2 `Asn1BigInteger (BigInteger value)`

This constructor assigns a big integer object.

Parameters

value `BigInteger` value

3.2.2.3 `Asn1BigInteger (System.String value)`

This constructor creates a new big integer object and sets it to the string value passed in. String value may contain the prefix that describes the radix: 0x - hexadecimal, 0o - octal, 0b - binary. The string value without prefix assumes decimal value. The optional sign '-' may be specified at the beginning of the string to specify the negative value.

Parameters

value String value

3.2.2.4 `Asn1BigInteger (System.String value, int radix)`

This constructor creates a new big integer object and sets it to the string value passed in. String value may contain the prefix that describes the radix: 0x - hexadecimal, 0o - octal, 0b - binary. The string value without prefix assumes decimal value. The optional sign '-' may be specified at the beginning of the string to specify the negative value.

Parameters

value String value

radix Can be 16 for hexadecimal, 8 for octal, 2 for binary or 10 for decimal

3.2.3 Member Function Documentation

3.2.3.1 `virtual void Decode (Asn1JsonDecodeBuffer buffer) [virtual]`

Decode ASN.1 INTEGER from JSON.

3.2.3.2 **override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]**

Decode this value as if unconstrained. Subclasses may override this method to decode according to the constraints on the respective ASN.1 type.

Reimplemented from [Asn1Type](#).

3.2.3.3 **void Decode (Asn1PerDecodeBuffer *buffer*, BigInteger *lower*, BigInteger *upper*)**

This method decodes an ASN.1 integer value using Packed Encoding Rules (PER). The length and contents components of the message are decoded. The decoded result is stored in the public `mValue` member variable.

Parameters

buffer PER Decode message buffer object

lower The PER-visible lower bound; null if there is not a lower bound.

upper The PER-visible upper bound; null if there is not an upper bound.

3.2.3.4 **override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]**

This method decodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are decoded. The decoded result is stored in the public `mValue` member variable.

Parameters

buffer PER Decode message buffer object

Reimplemented from [Asn1Type](#).

3.2.3.5 **override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]**

This method decodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.2.3.6 **void DecodeSigned (Asn1OerDecodeBuffer *buffer*, int *octets*)**

Decode a signed integer (2's complement form), of the given length, into this object.

Parameters

octets The number of octets in which the value was encoded.

3.2.3.7 void DecodeSigned (Asn1OerDecodeBuffer *buffer*)

Decode a variable-size signed integer, encoded with a length, into this object, following OER.

3.2.3.8 void DecodeUnsigned (Asn1OerDecodeBuffer *buffer*, int *octets*)

Decode an unsigned integer (a binary integer, not 2's complement form), of the given length, into this object.

Parameters

octets The number of octets in which the value was encoded.

3.2.3.9 void DecodeUnsigned (Asn1OerDecodeBuffer *buffer*)

Decode a variable-size unsigned integer, encoded with a length, into this object, following OER.

3.2.3.10 BigInteger DecodeValue (Asn1DecodeBuffer *buffer*, int *length*)

This method decodes the contents of an ASN.1 integer value using either the Basic Encoding Rules (BER) or the Packed Encoding Rules (PER).

Parameters

buffer Decode message buffer object

length Length of encoded contents

Returns

Decoded integer value

3.2.3.11 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 integer value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.2.3.12 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 integer value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.2.3.13 **override void Encode (Asn1PerOutputStream *outs*) [virtual]**

This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.2.3.14 **override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]**

This method encodes and writes to the stream an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, exception thrown by the underlying System.IO.Stream object.

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.2.3.15 **virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]**

Encode this ASN.1 INTEGER value to JSON

3.2.3.16 **override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]**

This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard.

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace prefix name

Reimplemented from [Asn1Type](#).

3.2.3.17 **override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]**

This method encodes an ASN.1 integer value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.2.3.18 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

Encode this value as if unconstrained. Subclasses may override this method to encode it according to the constraints on the respective ASN.1 type.

Reimplemented from [Asn1Type](#).

3.2.3.19 **void Encode (Asn1PerEncodeBuffer *buffer*, BigInteger *lower*, BigInteger *upper*)**

This method encodes an ASN.1 integer value using Packed Encoding Rules (PER).

The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

lower The PER-visible lower bound; null if there is not a lower bound.

upper The PER-visible upper bound; null if there is not an upper bound.

Returns

Length of component or negative status value

3.2.3.20 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.2.3.21 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object
explicitTagging Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.2.3.22 **override void EncodeAttribute (Asn1XmlEncoder *buffer*, String *attrName*)**

This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard.

Parameters

buffer Encode message buffer object
elemName Element name
attribute Attribute name

3.2.3.23 **void EncodeSigned (Asn1OerEncodeBuffer *buffer*)**

Encode this value as a variable-size signed integer, with length, according to OER.

3.2.3.24 **void EncodeSigned (Asn1OerEncodeBuffer *buffer*, int *octets*)**

Encode integer value as a signed value (2's complement) in the given number of octets.
The value must fit in the given number of octets.

Parameters

buffer The buffer.
octets The number of octets to encode in; $0 < \text{octets} \leq 8$

3.2.3.25 **void EncodeUnsigned (Asn1OerEncodeBuffer *buffer*, int *octets*)**

Encode integer value as an unsigned value (binary integer) in the given number of octets.
The value must be non-negative and fit in the given number of octets.

Parameters

buffer The buffer.
octets The number of octets to encode in; $0 < \text{octets} \leq 8$.

3.2.3.26 void EncodeUnsigned (Asn1OerEncodeBuffer *buffer*)

Encode this value as a variable-size unsigned integer, with length, according to OER.

3.2.3.27 static internal int EncodeValue (Asn1EncodeBuffer *buffer*, BigInteger *ivalue*, bool *doCopy*) [static, protected]

This method encodes an ASN.1 integer value's contents in accordance with either the ASN.1 Basic Encoding Rules (BER) or Packed Encoding Rules (PER).

Parameters

buffer Encode message buffer object

ivalue Integer value to encode

doCopy Encode the copy of the value

Returns

Length of encoded component

3.2.3.28 override bool Equals (System.Object *value*)

This method compares this value to the given [Asn1BigInteger](#) value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1BigInteger](#).

Returns

true if the specified Object is equal to the current Object; otherwise, false.

3.2.3.29 virtual bool Equals (long *value*) [virtual]

This method compares this value to the given integer value for equality.

Parameters

value The long value to compare with the current Object.

Returns

true if the specified long value is equal to the current Object; otherwise, false.

3.2.3.30 override int GetHashCode ()

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.2.3.31 override System.String ToString ()

This method will return a string representation of the integer value. The format is the ASN.1 value format for this type..

Returns

Stringified representation of the value

3.2.4 Member Data Documentation

3.2.4.1 new readonly Asn1Tag _TAG = new Asn1Tag(Asn1Tag.UNIV, Asn1Tag.PRIM, Asn1Type.INTEGER) [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 2).

Reimplemented from [Asn1Type](#).

3.2.4.2 BigInteger mValue

The magnitude of this [Asn1BigInteger](#), in *big-endian* order: the zeroth element of this array is the most-significant int of the magnitude. The magnitude must be "minimal" in that the most-significant int (`mValue[0]`) must be non-zero. This is necessary to ensure that there is exactly one representation for each [Asn1BigInteger](#) value. Note that this implies that the [Asn1BigInteger](#) zero has a zero-length mValue array.

3.3 Asn1BitString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Types

- enum [StringFormat](#)

Public Member Functions

- [Asn1BitString](#) (System.Collections.BitArray bitArray)
- [Asn1BitString](#) (System.String value_)
- [Asn1BitString](#) (bool[] bitValues)
- [Asn1BitString](#) (byte[] data)
- [Asn1BitString](#) (int numbits_, byte[] data)
- [Asn1BitString](#) ()
- virtual void [Clear](#) (int bitno)
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- void [Decode](#) (Asn1MderDecodeBuffer buffer, int length)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [DecodeContent](#) (Asn1OerDecodeBuffer buffer, int numOctets, int unusedBits)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix, System.String[] namedbits, int[] namedbitindex)
- virtual void [Encode](#) (Asn1XerEncoder buffer, System.String elemName, System.String[] namedbits, int[] namedbitindex)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- void [Encode](#) (Asn1MderOutputStream outs, int length)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- void [EncodeContent](#) (Asn1OerEncodeBuffer buffer)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (int nbits, byte[] value)
- virtual bool [Get](#) (int bitno)
- override int [GetHashCode](#) ()
- virtual bool [IsNamedBitStr](#) (System.String buffer)
- virtual void [Set](#) (int bitno)
- virtual void [Set](#) (int bitno, bool value)
- virtual bool[] [ToBoolArray](#) ()

- virtual System.String [ToHexString](#) ()
- virtual System.IO.Stream [toInputStream](#) ()
- override System.String [ToString](#) ()

Public Attributes

- byte[] [mValue](#)
- int [numbits](#)
- bool [trimZeroBits](#)

Static Public Attributes

- static new readonly [Asn1Tag _TAG](#)
- static [StringFormat mStringFormat](#) = StringFormat.HEXBIN

Protected Member Functions

- virtual int [GetOerEffectiveMin](#) ()

Properties

- override int [Length](#) [get]
- virtual bool [this](#) [int bitno] [get, set]

3.3.1 Detailed Description

This is a container class for holding the components of an ASN.1 bit string value.

3.3.2 Member Enumeration Documentation

3.3.2.1 enum StringFormat

Defines possible string fomrats.

The `HEX` constant describes the string format as hex digit value (e.g. 0123456789ABCDEF).

The `BITS` constant describes the string format as binary digit value (e.g. only 0 and 1 digits).

The `ASN1VALUE` constant describes the string format as hex or binary digit value. The binary string value will end with letter 'B' and hex string value will end with letter 'H' (e.g. '0101'B or '11'H). If the number of bits is less than or equal to 16, than it will be printed as Binary digits, else as Hex digits.

3.3.3 Constructor & Destructor Documentation

3.3.3.1 Asn1BitString ()

This constructor creates an empty bit string that can be used in a Decode method call to receive a bit string value.

3.3.3.2 Asn1BitString (int *numbits_*, byte[] *data*)

This constructor initializes a bit string with the given number of bits and data.

Parameters

numbits_ Number of bits

data Binary bit string contents

3.3.3.3 Asn1BitString (byte[] *data*)

This constructor initializes a bit string with the given bytes, using all 8 bits of each byte.

Parameters

data Binary bit string contents

3.3.3.4 Asn1BitString (bool[] *bitValues*)

This constructor initializes a bit string from the given boolean array. Each array position corresponds to a bit in the bit string.

Parameters

bitValues The boolean array

3.3.3.5 Asn1BitString (System.String *value_*)

This constructor parses the given ASN.1 value text (either a binary or hex data string) and assigns the values to the internal bit string.

Examples of valid value formats are as follows:

Binary string: '11010010111001'B

Hex string: '0fa56920014abc'H

Parameters

value_ The ASN.1 value specification text

3.3.3.6 Asn1BitString (System.Collections.BitArray *bitArray*)

This constructor initializes a bit string from the given BitSet object.

Parameters

bitArray C# BitArray object

3.3.4 Member Function Documentation

3.3.4.1 virtual void Clear (int *bitno*) [virtual]

This method clears the given bit in the bit string.

Parameters

bitno The zero-based index of the bit to clear. The bit numbers start at zero and with the MSB of the first byte and progress from left to right.

3.3.4.2 virtual void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 bit string from JSON.

3.3.4.3 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 BIT STRING that was encoded according to OER.

This assumes the BIT STRING was not fixed-size constrained, so that the length and unused bits are in the encoding. Subclasses can override this method to simply call DecodeContent for fixed-size constrained BIT STRINGS.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.3.4.4 void Decode (Asn1MderDecodeBuffer *buffer*, int *length*)

Decode a BIT STRING from the MDER encoding into this object. Exactly the given length of bits will be decoded.

Parameters

length This should be 8, 16, or 32, as these are the only lengths MDER supports. However, this is not checked here as it should be able to be checked at code generation time.

3.3.4.5 virtual void Decode (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER).

Parameters

buffer Decode message buffer object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

3.3.4.6 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 bit string value using the packed encoding rules (PER). The string is assumed to not contain a size constraint.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.3.4.7 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 bit string value using the BER or DER encoding rules. The UNIVERSAL tag value and length are decoded if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.3.4.8 void DecodeContent (Asn1OerDecodeBuffer *buffer*, int *numOctets*, int *unusedBits*)

<summary> This method decodes an ASN.1 BIT STRING's content that was encoded according to OER, given the total number of octets and the number of unused bits in the last octet.

</summary>

Parameters

buffer Decode message buffer object

numOctets Total number of octets encoding the content, including the final, partial octet (if *unusedBits* > 0).

unusedBits # of unused bits in last octet.

3.3.4.9 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes ASN.1 bit string type using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.3.4.10 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes ASN.1 bit string type using the XML decoding as specified in the XML Schema standard.

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.3.4.11 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, long *upper*) [virtual]

This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER) into the stream. The value to be encoded is stored in the 'numbits' and 'mValue' public member variables within this class.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Output Stream object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.3.4.12 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an unconstrained ASN.1 bit string value using the packed encoding rules (PER) into the stream. The value to be encoded is stored in the 'numbits' and 'mValue' public member variables within this class.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Output Stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.3.4.13 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 bit string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, exception thrown by the underlying System.IO.Stream object.

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.3.4.14 virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode this bit string to JSON.

Parameters

outstream

3.3.4.15 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes ASN.1 bit string type using the XML Encoding as specified in the XML Schema standard.

Parameters

buffer Encode message buffer object

elemName XML element name used to wrap string

nsPrefix XML element namespace name

Reimplemented from [Asn1Type](#).

3.3.4.16 virtual void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*, System.String[] *namedbits*, int[] *namedbitindex*) [virtual]

This method encodes ASN.1 bit string type using the XML Encoding as specified in the XML Schema standard.

Parameters

buffer Encode message buffer object

elemName XML element name used to wrap string

nsPrefix XML element namespace name

namedbits Array of named bits

namedbitindex Array of named bits index values

3.3.4.17 virtual void Encode (Asn1XerEncoder *buffer*, System.String *elemName*, System.String[] *namedbits*, int[] *namedbitindex*) [virtual]

This method encodes ASN.1 bit string type using the XML Encoding as specified in the itu-t XER standard.

Parameters

buffer Encode message buffer object

elemName XML element name used to wrap string

namedbits Array of named bits

namedbitindex Array of named bits index values

3.3.4.18 **override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]**

This method encodes ASN.1 bit string type using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object
elemName XML element name used to wrap string

Reimplemented from [Asn1Type](#).

3.3.4.19 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

This method encodes this ASN.1 BIT STRING value, according to OER. This encodes the length determinant and # of unused bits, assuming that the BIT STRING is not fixed-size constrained. Subclasses may override this to simply call EncodeContent for fixed-size constrained strings.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.3.4.20 **void Encode (Asn1MderOutputStream *outs*, int *length*)**

Encode this BIT STRING into the MDER encoding. The length of this BIT STRING must match the given length.

Parameters

length This should be 8, 16, or 32, as these are the only lengths MDER supports. However, this is not checked here as it should be able to be checked at code generation time. We only check here that the actual and given lengths match.

3.3.4.21 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]**

This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER). The value to be encoded is stored in the 'numbits' and 'mValue' public member variables within this class.

Parameters

buffer Encode message buffer object
lower Lower bound (inclusive) of size constraint
upper Upper bound (inclusive) of size constraint

3.3.4.22 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an unconstrained ASN.1 bit string value using the packed encoding rules (PER). The value to be encoded is stored in the 'numbits' and 'mValue' public member variables within this class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.3.4.23 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 bit string value using the BER or DER encoding rules. The UNIVERSAL tag value and length are encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object
explicitTagging Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.3.4.24 **void EncodeContent (Asn1OerEncodeBuffer *buffer*)**

This method encodes the content of an ASN.1 bit string value, according to OER. (The length determinant and # of unused bits is not encoded.)

Parameters

buffer Encode message buffer object

3.3.4.25 **override bool Equals (System.Object *value*)**

This method compares this bit string value to the given value for equality. This method assumes all unused bits in the last byte are set to zero.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1BitString](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.3.4.26 **virtual bool Equals (int *nbits*, byte[] *value*) [virtual]**

This method compares this bit string value to the given value for equality. This method assumes all unused bits in the last byte are set to zero.

Parameters

nbits The number of bits to compare from the byte array.
value The byte array to compare with the current Object.

Returns

`true` if the specified bit array is equal to the current Object; otherwise, `false`.

3.3.4.27 virtual bool Get (int *bitno*) [virtual]

Gets the value of the bit at a specific position in the bit array.

Parameters

bitno The zero-based index of the bit to get. The bit numbers start at zero and with the MSB of the first byte and progress from left to right.

Returns

true if bit is set; otherwise false.

3.3.4.28 override int GetHashCode ()

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.3.4.29 virtual int GetOerEffectiveMin () [protected, virtual]

Return the lower bound for the OER effective size constraint. When trimZeroBits is set, this controls the trimming done for OER encoding.

Subclasses must override this to return the correct value if the BIT STRING has named bits and the effective size constraint has a lower bound other than zero.

3.3.4.30 virtual bool IsNamedBitStr (System.String *buffer*) [virtual]

This method determines is the input character string represented as named bit string or as bits sequence. It is used for XML decoding.

Parameters

buffer Bit string as string to be tested.

Returns

true, if bit string is represented as sequenece of identifiers.

3.3.4.31 virtual void Set (int *bitno*) [virtual]

This method will set the given bit number to one (1). It will expand the existing bit array if it needs to.

Parameters

bitno The zero-based index of the bit to set. The bit numbers start at zero and with the MSB of the first byte and progress from left to right.

3.3.4.32 virtual void Set (int *bitno*, bool *value*) [virtual]

This method sets the given bit number in the bit string with given value. It will expand the existing bit array if it needs to.

Parameters

bitno The zero-based index of the bit to set. The bit numbers start at zero and with the MSB of the first byte and progress from left to right.

value The Boolean value to assign to the bit.

3.3.4.33 virtual bool [] ToBoolArray () [virtual]

This method converts the bit string stored in this object to a boolean array.

Returns

Boolean array value

3.3.4.34 virtual System.String ToHexString () [virtual]

This method will return a hex string representation of the bit string value. The output format is a string of hex bytes with no extra delimiting characters (ex. 0D56EF).

Returns

Stringified representation of the value

3.3.4.35 virtual System.IO.Stream toInputStream () [virtual]

This method will return a byte array input stream representation of the bit string value.

Returns

Reference to System.IO.MemoryStream object

3.3.4.36 override System.String ToString ()

This method will return a string representation of the bit string value. The output format is a string of hex digits/binary digits according to the value set for **mStringFormat** variable.

Returns

String representation of the value

3.3.5 Member Data Documentation

3.3.5.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 3).

Reimplemented from [Asn1Type](#).

3.3.5.2 `StringFormat mStringFormat = StringFormat.HEXBIN` `[static]`

The `mStringFormat` variable can be used to set the string format for `print()` or event handler calls or `toString()` functions. The possible options are: `HEX`, `BITS`, `ASN1VALUE` `HEX` is the default format.

3.3.5.3 `byte [] mValue`

This variable holds the bit string value. These are the bits that are encoded when `encode` is invoked. It is also where the decoded bit string is stored after a `Decode` operation.

3.3.5.4 `int numbits`

This variable contains the number of bits in the bit string value.

3.3.5.5 `bool trimZeroBits`

This variable describes whether trailing zero bits should be truncated. This is required when encoding a named bit string using `CER`, `DER`, `PER`, or `C-OER`. By default, no trimming is done.

3.3.6 Property Documentation

3.3.6.1 `override int Length` `[get]`

Gets the length of the BIT STRING in bits.

Value: Number of bits.

Reimplemented from [Asn1Type](#).

3.3.6.2 `virtual bool this[int bitno]` `[get, set]`

Gets or Sets the given bit in the bit string. It will expand the existing bit array if it needs to set the bit value.

Parameters

bitno The position of the bit in bit array

Value: true if bit is set; otherwise false.

3.4 Asn1BMPString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1CharString](#).

Public Member Functions

- [Asn1BMPString](#) (System.String data)
- [Asn1BMPString](#) ()
- void [Decode](#) (Asn1OerDecodeBuffer buffer, int length)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1OerEncodeBuffer buffer, bool withLength)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Public Attributes

- const int [BITSPERCHAR](#) = 16

Static Public Attributes

- static new readonly [Asn1Tag _TAG](#)

3.4.1 Detailed Description

This is a container class for holding the components of an ASN.1 BMP string value.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Asn1BMPString ()

The default constructor creates an empty string object.

3.4.2.2 Asn1BMPString (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string value.

Parameters

data string representation of BMPString

3.4.3 Member Function Documentation

3.4.3.1 void Decode (Asn1OerDecodeBuffer *buffer*, int *length*)

Decode the value in accordance with OER.

This class's implementation decodes a string of the given length. This method is non-virtual as I don't see any reason for overriding it.

Parameters

length Length of string to decode, in characters.

3.4.3.2 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode the value in accordance with OER.

This class's implementation decodes the string with a length determinant. If a subclass should be decoded without a length determinant, it should override this method to invoke decode(buffer, length). Subclasses may override this to add constraint checks after invoking this method to decode the string.

Reimplemented from [Asn1Type](#).

3.4.3.3 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*) [virtual]

This overloaded version of the Decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Decode message buffer object

charSet Object representing permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.4.3.4 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but allows a permitted alphabet character set to be specified. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Decode message buffer object

charSet Object representing permitted alphabet constraint character set (optional)

3.4.3.5 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.4.3.6 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 BMP string value including the UNIVERSAL tag value and length if explicit tagging is specified. This string type uses 16-bit characters.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.4.3.7 virtual void Encode (Asn1PerOutputStream *outs*, Asn1CharSet *charSet*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 BMP string value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the underlying [Asn1PerOutputStream](#).

Parameters

outs PER Output Stream object

charSet Object representing the permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.4.3.8 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet) [virtual]

This method encodes an ASN.1 BMP string value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable in the `Asn1CharString` base class.

Also throws any exception thrown by the underlying `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

charSet Object representing permitted alphabet constraint character set (optional)

Exceptions

Asn1Exception Thrown, if operation is failed.

3.4.3.9 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an ASN.1 BMP string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the `Asn1CharString` base class.

Also throws any exception thrown by the underlying `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from `Asn1Type`.

3.4.3.10 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 BMP string including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, exception thrown by the underlying `System.IO.Stream` object.

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from `Asn1Type`.

3.4.3.11 virtual void Encode (Asn1OerEncodeBuffer *buffer*, bool *withLength*) [virtual]

Encode the string, with or without a length determinant.

Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters

withLength true if a length determinant should be encoded.

3.4.3.12 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

Encode the value in accordance with OER.

This class's implementation invokes encode(*buffer*, true) to encode the string with a length determinant. If a subclass should be encoded without a length determinant, it should override this to invoke encode(*buffer*, false).

Reimplemented from [Asn1Type](#).

3.4.3.13 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

charSet Object representing the permitted alphabet constraint character set

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.4.3.14 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Encode message buffer object

charSet Object representing permitted alphabet constraint character set (optional)

3.4.3.15 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.4.3.16 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.4.4 Member Data Documentation

3.4.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 30).

Reimplemented from [Asn1Type](#).

3.4.4.2 const int BITSPERCHAR = 16

The BITSPERCHAR constant specifies the number of bits per character for PER (aligned or unaligned).

3.5 Asn1Boolean Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1Boolean](#) (bool value_)
- [Asn1Boolean](#) ()
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- void [Encode](#) (Asn1XmlEncoder buffer, String elemName, String nsPrefix, bool asText)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XerEncodeBuffer buffer)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (bool value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

Static Public Member Functions

- static void [setTrueEncodedByte](#) (byte b)

Public Attributes

- bool [mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)
- static readonly [Asn1Boolean FALSE_VALUE](#) = new [Asn1Boolean](#)(false)
- static readonly [Asn1Boolean TRUE_VALUE](#) = new [Asn1Boolean](#)(true)

3.5.1 Detailed Description

This class represents the ASN.1 BOOLEAN built-in type.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Asn1Boolean ()

The default constructor sets the boolean value to false.

3.5.2.2 Asn1Boolean (bool *value_*)

This constructor creates a boolean object from a boolean value.

Parameters

value_ Boolean value

3.5.3 Member Function Documentation

3.5.3.1 virtual void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 BOOLEAN from JSON.

3.5.3.2 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode BOOLEAN value according to OER.

Reimplemented from [Asn1Type](#).

3.5.3.3 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 boolean value using

the Packed Encoding Rules (PER).

The decoded result is stored in the public member `mValue`

in this object.

Parameters

buffer PER Decode message buffer object

Reimplemented from [Asn1Type](#).

3.5.3.4 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 boolean value including the UNIVERSAL

tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER) or the Distinguished Encoding Rules (DER).

The decoded result is stored in the public member `mValue`

in this object.

Parameters

buffer Decode message buffer object
explicitTagging Flag indicating element is explicitly tagged
implicitLength Length of contents if implicit

Returns

Decoded boolean value

Reimplemented from [Asn1Type](#).

3.5.3.5 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 boolean value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded
attrs Attributes string from element tag

3.5.3.6 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 boolean value using the XML Schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded
attrs Attributes string from element tag

3.5.3.7 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 boolean value using the Packed Encoding Rules (PER).

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Encode message stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.5.3.8 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 boolean value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, exception thrown by the underlying System.IO.Stream object.

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.5.3.9 virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode this boolean value to JSON.

3.5.3.10 void Encode (Asn1XmlEncoder *buffer*, String *elemName*, String *nsPrefix*, bool *asText*)

This method encodes an ASN.1 boolean value. It is for use with extended-XER.

Parameters

buffer Encode message buffer object

elemName Element name for optional surrounding element.

nsPrefix XML element name space prefix

asText If true, encode as text. Otherwise, encode as an empty element.

3.5.3.11 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an ASN.1 boolean value according to the Obj-Sys XML encoding rules. It encodes the value as XML text.

Parameters

buffer Encode message buffer object

elemName Element name for optional surrounding element.

nsPrefix Element namespace prefix value

Reimplemented from [Asn1Type](#).

3.5.3.12 virtual void Encode (Asn1XerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 boolean value using the XML encoding rules (XER). This method does not add start and end tags (<tag> and </tag>), only value is encoded (<true/> or <false/>).

Parameters

buffer Encode message buffer object

3.5.3.13 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes an ASN.1 boolean value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.5.3.14 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

Encode BOOLEAN value according to OER.

Reimplemented from [Asn1Type](#).

3.5.3.15 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 boolean value using the Packed Encoding Rules (PER).

Parameters

buffer PER Encode message buffer object

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.5.3.16 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 boolean value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER) or the Distinguished Encoding Rules (DER).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length (in octets) of encoded component

Reimplemented from [Asn1Type](#).

3.5.3.17 override void EncodeAttribute (Asn1XmlEncoder *buffer*, System.String *attrName*) [virtual]

This method encodes an ASN.1 boolean value using the XML Encoding as specified in the W3C XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

attrName Element name

Reimplemented from [Asn1Type](#).

3.5.3.18 override bool Equals (System.Object value)

This method compares this boolean value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1Boolean](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.5.3.19 virtual bool Equals (bool value) [virtual]

This method compares this boolean value to the given value for equality.

Parameters

value The bool value to compare with the current Object.

Returns

`true` if the specified bool value is equal to the current Object; otherwise, `false`.

3.5.3.20 override int GetHashCode ()

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.5.3.21 static void setTrueEncodedByte (byte b) [static]

This method sets the byte value that represents the boolean value TRUE. If a zero byte (0x00) is passed, the value is transparently set to 0xFF, the valid representation for BER, CER, and DER.

Parameters

b The byte value to set.

3.5.3.22 **override System.String ToString ()**

This method will return a string representation of the boolean value. The format is the ASN.1 value format for this type.

Returns

Stringified representation of the value

3.5.4 **Member Data Documentation**

3.5.4.1 **new readonly Asn1Tag _TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 1).

Reimplemented from [Asn1Type](#).

3.5.4.2 **readonly Asn1Boolean FALSE_VALUE = new Asn1Boolean(false) [static]**

The `FALSE_VALUE` constant represents a boolean FALSE value.

3.5.4.3 **bool mValue**

This public member variable is where the boolean value is stored. This is the value that is encoded when one of the encode methods is called. It is also where the decoded result is stored when a Decode method is called.

3.5.4.4 **readonly Asn1Boolean TRUE_VALUE = new Asn1Boolean(true) [static]**

The `TRUE_VALUE` constant represents a boolean TRUE value.

3.6 Asn1CharRange Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1CharSet](#).

Public Member Functions

- [Asn1CharRange](#) (int lower, int upper)
- override int [GetCharAtIndex](#) (int index)
- override int [GetCharIndex](#) (int charValue)
- override bool [validate](#) (String s)

Protected Attributes

- internal int [mLower](#)
- internal int [mUpper](#)

Properties

- override int [MaxValue](#) [get]

3.6.1 Detailed Description

This class is used to represent a permitted alphabet that is specified

as a large, continuous range of characters. An example of this can be found in the following extract from T.124:

```
simpleTextFirstCharacter UniversalString ::= {0, 0, 0, 0}
```

```
simpleTextLastCharacter UniversalString ::= {0, 0, 0, 255}
```

```
SimpleTextString ::= BMPString (SIZE (0..255))
```

```
(FROM (simpleTextFirstCharacter..simpleTextLastCharacter))
```

This class is mainly for internal use by the compiler when generating

methods that encode/Decode PER character string components containing permitted alphabet constraints.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Asn1CharRange (int lower, int upper)

This constructor sets the range values.

Parameters

- lower* Range lower value
- upper* Range upper value

3.6.3 Member Function Documentation

3.6.3.1 override int GetCharAtIndex (int *index*) [virtual]

This method will fetch the character from the permitted alphabet at the given index.

Parameters

index Index of character within the character set

Returns

Character at given index

Exceptions

[Asn1ConsVioException](#) Index not within define range

Implements [Asn1CharSet](#).

3.6.3.2 override int GetCharIndex (int *charValue*) [virtual]

This method will determine the index of the given character within the permitted alphabet character set.

Parameters

charValue Character value to search for

Returns

Index of character

Exceptions

[Asn1ConsVioException](#) Character not found in set

Implements [Asn1CharSet](#).

3.6.3.3 override bool validate (String *s*) [virtual]

This method will validate a character string by comparing its contents to the character range. Each character in the string is checked against the range. If it exceeds the upper or lower limit of the range, false is returned. Otherwise true is returned.

Parameters

s The string to be validated.

Returns

False if the string contains invalid characters; true otherwise.

Implements [Asn1CharSet](#).

3.6.4 Member Data Documentation

3.6.4.1 internal int mLower [protected]

This variable represents the lower value of the range.

3.6.4.2 internal int mUpper [protected]

This variable represents the upper value of the range.

3.6.5 Property Documentation

3.6.5.1 override int MaxValue [get]

Gets the maximum value of character within the given permitted alphabet character set.

Value: Upper Bound Character or Character with max int value

Reimplemented from [Asn1CharSet](#).

3.7 Asn1CharSet Class Reference

Inherited by [Asn1BMPStringCharset](#), [Asn1CharRange](#), [Asn1DiscreteCharset](#), [Asn1IA5StringCharset](#), [Asn1NumericStringCharset](#), [Asn1PrintableStringCharset](#), and [Asn1VisibleStringCharset](#).

Public Member Functions

- abstract int [GetCharAtIndex](#) (int index)
- abstract int [GetCharIndex](#) (int charValue)
- virtual int [GetNumBitsPerChar](#) (bool aligned)
- abstract bool [validate](#) (String s)

Protected Member Functions

- internal [Asn1CharSet](#) (int nchars)

Protected Attributes

- internal int [mABitsPerChar](#)
- internal int [mUBitsPerChar](#)

Properties

- abstract int [MaxValue](#) [get]

3.7.1 Detailed Description

This is the base class for representing character sets that are defined in ASN.1 permitted alphabet constraints.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 internal Asn1CharSet (int nchars) [protected]

This constructor sets the number of bits-per-character values based on the given number of characters in the character set.

Parameters

nchars Number of characters in the character set

3.7.3 Member Function Documentation

3.7.3.1 abstract int GetCharAtIndex (int index) [pure virtual]

This method will fetch the character from the permitted alphabet at the given index.

Parameters

index Index of character within the character set

Returns

Character at given index

Exceptions

Asn1ConsVioException Index not within define range

Implemented in [Asn1CharRange](#), and [Asn1DiscreteCharSet](#).

3.7.3.2 abstract int GetCharIndex (int *charValue*) [pure virtual]

This method will determine the index of the given character within the permitted alphabet character set.

Parameters

charValue Character value to search for

Returns

Index of character

Exceptions

Asn1ConsVioException Character not found in set

Implemented in [Asn1CharRange](#), and [Asn1DiscreteCharSet](#).

3.7.3.3 virtual int GetNumBitsPerChar (bool *aligned*) [virtual]

This method will return the number of bits-per-character.

Parameters

aligned Boolean value indicating whether number of aligned (true) or unaligned (false) characters should be returned.

Returns

Number of bits-per-character

3.7.3.4 abstract bool validate (String *s*) [pure virtual]

This method will validate a character string by comparing its contents to the character set. If a character string contains characters that are not in the character set, this method will return false. Otherwise it returns true.

Parameters

s The string to be validated.

Returns

False if the string contains invalid characters; true otherwise.

Implemented in [Asn1CharRange](#), and [Asn1DiscreteCharSet](#).

3.7.4 Member Data Documentation

3.7.4.1 internal int mABitsPerChar [protected]

This variable holds number of bits-per-character (PER aligned).

3.7.4.2 internal int mUBitsPerChar [protected]

This variable holds number of bits-per-character (PER unaligned).

3.7.5 Property Documentation

3.7.5.1 abstract int MaxValue [get]

Gets the maximum value of the character within the given permitted alphabet character set.

Value: Upper Bound Character or Character with max int value

Reimplemented in [Asn1CharRange](#), and [Asn1DiscreteCharSet](#).

3.8 Asn1CharString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Inherited by [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1UTF8String](#), and [Asn1VarWidthCharString](#).

Public Member Functions

- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- void [DecodeByteToChar](#) (Asn1OerDecodeBuffer buffer, int length)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1JsonOutputStream outs)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override bool [Equals](#) (System.Object value)
- bool [Equals](#) (System.String value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()
- bool [validate](#) ([Asn1CharSet](#) charSet)

Public Attributes

- System.String [mValue](#)

Protected Member Functions

- internal [Asn1CharString](#) (System.String data, short typeCode)
- internal [Asn1CharString](#) (short typeCode)
- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength, [Asn1Tag](#) tag)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging, [Asn1Tag](#) tag)

Static Protected Member Functions

- static int [Encode](#) (Asn1BerEncodeBuffer buffer, String val, bool explicitTagging, [Asn1Tag](#) tag)

Protected Attributes

- internal System.Text.StringBuilder [mStringBuffer](#)

Properties

- override int [Length](#) [get]

3.8.1 Detailed Description

This is a container class for holding the components of an ASN.1 character string value. Subclasses are defined for all of the different string types.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 internal Asn1CharString (short *typeCode*) [protected]

This constructor creates an empty string that can be used in a Decode method call to receive a string value.

Parameters

typeCode Universal ID code for ASN.1 character string

3.8.2.2 internal Asn1CharString (System.String *data*, short *typeCode*) [protected]

This constructor initializes a character string from the given string data.

Parameters

data Character string

typeCode Universal ID code for ASN.1 character string

3.8.3 Member Function Documentation

3.8.3.1 virtual void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 restricted character string from JSON.

3.8.3.2 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]

This overloaded version of the Decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable.

Parameters

buffer Decode message buffer object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.8.3.3 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [protected, virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes that a permitted alphabet constraint has been specified that would reduce the number of bits-per-character from the default character set. It also assumes a general length determinant is present (i.e. there is not size constraint). The decoded result is stored in the public `mValue` member variable.

Parameters

- buffer* Decode message buffer object
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)

3.8.3.4 virtual internal void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*, Asn1Tag *tag*) [protected, virtual]

This method decodes an ASN.1 character string value including the UNIVERSAL tag value and length if explicit tagging is specified. It is a protected method that can only be accessed by objects subclassed from this type.

Parameters

- buffer* Decode message buffer object
- explicitTagging* Flag indicating element is explicitly tagged
- implicitLength* Length of contents if implicit
- tag* Universal tag to apply

3.8.3.5 void DecodeByteToChar (Asn1OerDecodeBuffer *buffer*, int *length*)

This method decodes a given number of bytes and converts each byte to a char having the same value.

3.8.3.6 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. using the XML encoding rules (XER).

Parameters

- buffer* String containing data to be decoded
- attrs* Attributes string from element tag

3.8.3.7 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. using the XML Schema encoding rules(ASN2XSD).

Parameters

- buffer* String containing data to be decoded
- attrs* Attributes string from element tag

3.8.3.8 virtual void Encode (Asn1JsonOutputStream *outs*) [virtual]

Encode the value of this object as a JSON string.

Parameters

out

3.8.3.9 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object
elemName XML element name used to wrap string
nsPrefix XML element namespace value

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.8.3.10 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object
elemName XML element name used to wrap string

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.8.3.11 virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]

This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object
abpc Number of bits per character (aligned)
ubpc Number of bits per character (unaligned)
charSet Object representing the permitted alphabet constraint character set (optional)
lower Effective size constraint lower bound
upper Effective size constraint upper bound

3.8.3.12 virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [protected, virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing the permitted alphabet constraint character set (optional)

3.8.3.13 virtual internal int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*, Asn1Tag *tag*) [protected, virtual]

This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified (the universal identifier must be provided by the caller).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Returns

Length in octets of encoded component

3.8.3.14 static int Encode (Asn1BerEncodeBuffer *buffer*, String *val*, bool *explicitTagging*, Asn1Tag *tag*) [static, protected]

This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified (the universal identifier must be provided by the caller).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Returns

Length in octets of encoded component

3.8.3.15 **override bool Equals (System.Object value)**

This method compares this character string value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1CharString](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.8.3.16 **bool Equals (System.String value)**

This method compares this character string value to the given value for equality.

Parameters

value The String value to compare with the current Object.

Returns

`true` if the specified string is equal to the current Object; otherwise, `false`.

3.8.3.17 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.8.3.18 **override System.String ToString ()**

This method will return a string representation of the value. The format is the ASN.1 value format for this type..

Returns

Stringified representation of the value

3.8.3.19 **bool validate (Asn1CharSet charSet)**

This method will attempt to validate a string against its internal character set.

Returns

True or False.

3.8.4 Member Data Documentation

3.8.4.1 internal System.Text.StringBuilder mStringBuffer [protected]

The `mStringBuffer` member variable is used to do internal operations on a string being encoded or decoded. Users should create it before using if it is null.

3.8.4.2 System.String mValue

The `mValue` public member variable is used to hold the string value to be encoded or the results of a Decode operation.

3.8.5 Property Documentation

3.8.5.1 override int Length [get]

Gets the length of the character string in characters.

Value: Number of characters.

Reimplemented from [Asn1Type](#).

3.9 Asn1Choice Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1Choice](#) ()
- override bool [Equals](#) (System.Object value)
- virtual [Asn1Type GetElement](#) ()
- override int [GetHashCode](#) ()
- virtual void [SetElement](#) (int choiceID, [Asn1Type](#) element)

Protected Attributes

- internal int [choiceID](#)
- internal [Asn1Type](#) [element](#)

Properties

- virtual int [ChoiceID](#) [get]
- abstract System.String [ElemName](#) [get]

3.9.1 Detailed Description

This class represents the ASN.1 CHOICE built-in type.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 [Asn1Choice](#) ()

The default constructor initializes the choiceID and value.

3.9.3 Member Function Documentation

3.9.3.1 override bool [Equals](#) (System.Object *value*)

This method compares this type element with the passed type element.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1Choice](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.9.3.2 virtual Asn1Type GetElement () [virtual]

This method returns the element object.

Returns

element data member.

3.9.3.3 override int GetHashCode ()

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.9.3.4 virtual void SetElement (int choiceID, Asn1Type element) [virtual]

This protected method sets the choice ID and value to the given values. Refer the Set_<element>() methods in compiler generated inherited classes.

Parameters

choiceID The identifier for element value. The identifier value can be defined by compiler-generated derived class.

element The element value. The possible value types can be defined by compiler-generated derived class.

3.9.4 Member Data Documentation

3.9.4.1 internal int choiceID [protected]

This member variable is where the selected choice option identifier is stored. This selects the choice option to be used. It is populated with one of the generated choice ID constants in a compiler-generated derived class.

3.9.4.2 internal Asn1Type element [protected]

This member variable is where the selected choice option value is stored. It can be accessed via the get and set methods in this class and in compiler-generated derived classes.

3.9.5 Property Documentation

3.9.5.1 virtual int ChoiceID [get]

Gets the choice identifier.

Value: choice option identifier

3.9.5.2 abstract System.String ElemName [get]

Gets the name of the selected element. A concrete version is generated by the compiler-generated derived class.

Value: choice option name

3.10 Asn1ChoiceExt Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1OpenType](#).

Public Member Functions

- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Public Attributes

- short [choiceIndex](#)
- [Asn1Tag tag](#) = null

3.10.1 Detailed Description

This is a container class for holding a CHOICE open type extension element. This class is used for an open type extension (i.e. a ... at the end of a constructed type or a ..., ... at some other point in a constructed type).

3.10.2 Member Function Documentation

3.10.2.1 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an open type extension in a CHOICE construct using the packed encoding rules (PER). This method will capture the extension item in an open type object and store it in the `mValue` public member list variable. The public member variable `choiceIndex` will be populated with the decoded choice index value.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1OpenType](#).

3.10.2.2 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an extension field using the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1OpenType](#).

3.10.2.3 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

Encode this unknown choice extension using the Octet Encoding Rules (OER). This uses this object's tag as the tag, and then encodes the value field as open type content.

Reimplemented from [Asn1OpenType](#).

3.10.2.4 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER).

Parameters

outs PER Output Stream object

Reimplemented from [Asn1OpenType](#).

3.10.2.5 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).

Parameters

outs BER Output Stream object

explicitTagging Flag indicating element is explicitly tagged

Reimplemented from [Asn1OpenType](#).

3.10.2.6 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1OpenType](#).

3.10.2.7 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating element is explicitly tagged

Returns

Length of encoded component

Reimplemented from [Asn1OpenType](#).

3.10.3 Member Data Documentation

3.10.3.1 short choiceIndex

The choice index value is used with the packed encoding rules (PER) when this object is used to encode/Decode a choice extension.

3.10.3.2 Asn1Tag tag = null

tag is used with OER. When decoding, it should be set to hold the decoded tag. When encoding, it is encoded as the tag for the unknown choice extension. The form (tag.mForm) is irrelevant.

3.11 Asn1ConsVioException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1ConsVioException](#) (System.String varname, System.String value)
- [Asn1ConsVioException](#) (System.String varname, double value)
- [Asn1ConsVioException](#) (System.String varname, long value)

3.11.1 Detailed Description

This class defines the 'ASN.1 constraint violation' exception that is thrown when an element is parsed that is outside the bounds to a defined constraint..

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Asn1ConsVioException (System.String *varname*, long *value*)

This constructor creates an exception object with a standard message based on the given variable name and value. The form of the message is "Element '*varname*' with value '*value*' violates defined constraint".

Parameters

varname Name of variable that violates constraint

value Value of variable

3.11.2.2 Asn1ConsVioException (System.String *varname*, double *value*)

This constructor creates an exception object with a standard message based on the given variable name and value. The form of the message is "Element '*varname*' with value '*value*' violates defined constraint".

Parameters

varname Name of variable that violates constraint

value Value of variable

3.11.2.3 Asn1ConsVioException (System.String *varname*, System.String *value*)

This constructor creates an exception object with a standard message based on the given variable name and value. The form of the message is "Element '*varname*' with value '*value*' violates defined constraint".

Parameters

varname Name of variable that violates constraint

value Value of variable

3.12 Asn1DecodeBuffer Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1MessageBuffer](#).

Public Member Functions

- virtual void [AddCaptureBuffer](#) (System.IO.MemoryStream buffer)
- virtual void [Capture](#) (int nbytes)
- virtual long [DecodeIntValue](#) (int length, bool signExtend)
- virtual int[] [DecodeOIDContents](#) (int llen)
- virtual int[] [DecodeRelOIDContents](#) (int llen)
- override System.IO.Stream [GetInputStream](#) ()
- virtual void [HexDump](#) ()
- virtual void [Mark](#) ()
- virtual void [Read](#) (byte[] buffer)
- virtual void [Read](#) (byte[] buffer, int offset, int nbytes)
- virtual int [Read](#) ()
- int [Read2Bytes](#) ()
- int [Read4Bytes](#) ()
- abstract int [ReadByte](#) ()
- virtual void [RemoveCaptureBuffer](#) (System.IO.MemoryStream buffer)
- virtual void [Reset](#) ()
- virtual void [SetInputStream](#) (byte[] msgdata, int offset, int length)
- virtual long [Skip](#) (long nbytes)

Protected Member Functions

- virtual internal void [Init](#) ()

Protected Attributes

- internal int [mByteCount](#)

Properties

- virtual int [ByteCount](#) [get]
- bool [LazyOpenTypeDecode](#) [get, set]

3.12.1 Detailed Description

This is the base class to specific Decode buffer classes for the different types of encoding rules (BER, DER and PER).

3.12.2 Member Function Documentation

3.12.2.1 virtual void AddCaptureBuffer (System.IO.MemoryStream *buffer*) [virtual]

This method is used to add a capture buffer to the internal capture buffer list. A capture buffer is used to capture all bytes read from this position forward from the input stream.

Parameters

buffer Buffer into which captured bytes are to be stored

3.12.2.2 virtual void Capture (int *nbytes*) [virtual]

This method captures bytes from the input stream to a separate object for later analysis.

Parameters

nbytes Number of bytes to capture

3.12.2.3 virtual long DecodeIntValue (int *length*, bool *signExtend*) [virtual]

This method decodes the contents of an ASN.1 integer value. It can be used for either BER or PER decoding.

Parameters

length Length of encoded contents

signExtend Sign-extend the decoded value to form a 2's comp result

Returns

Decoded long integer value

3.12.2.4 virtual int [] DecodeOIDContents (int *llen*) [virtual]

This method decodes the contents of an ASN.1 object identifier value. It can be used for either BER or PER decoding.

Parameters

llen Length of encoded contents

Returns

Decoded object identifier value

3.12.2.5 virtual int [] DecodeRelOIDContents (int *llen*) [virtual]

This method decodes the contents of an ASN.1 relative object identifier value. It can be used for either BER or PER decoding.

Parameters

llen Length of encoded contents

Returns

Decoded object identifier value

3.12.2.6 override System.IO.Stream GetInputStream () [virtual]

This method returns a reference to the current current Decode input stream object.

Returns

New input stream object containing encoded message

Implements [Asn1MessageBuffer](#).

3.12.2.7 virtual void HexDump () [virtual]

This method provides a hex dump of the bytes in the message being decoded.

3.12.2.8 virtual internal void Init () [protected, virtual]

This method initializes the input stream for decoding.

3.12.2.9 virtual void Mark () [virtual]

This method is used to mark the current position in the input stream for retry processing.

3.12.2.10 virtual void Read (byte[] *buffer*) [virtual]

This version of the read method reads the number of bytes equal to the length of the given input buffer.

Throws, Exception thrown by C# System.IO.Stream for I/O error, for Stream as input data

Parameters

buffer the buffer into which the data is read

Exceptions

[Asn1EndOfBufferException](#) Thrown if at end-of-stream

3.12.2.11 virtual void Read (byte[] *buffer*, int *offset*, int *nbytes*) [virtual]

This version of the read method reads the given number of bytes from the current input stream and writes them to the specified byte array at the given offset. It also writes the data to all registered capture buffers.

Throws, Exception thrown by C# System.IO.Stream for I/O error for Stream as input data

Parameters

buffer the buffer into which the data is read

offset the start offset of the data

nbytes number of bytes to read

Exceptions

Asn1EndOfBufferException Thrown if at end-of-stream

3.12.2.12 virtual int Read () [virtual]

The read method reads a single byte from the current input stream and returns it to the caller. It will also write the byte out to all registered capture buffers.

Throws, Exception thrown by C# System.IO.Stream for I/O error for Stream as input data

Returns

byte that was read from the input stream

Exceptions

Asn1EndOfBufferException Thrown if at end-of-stream

3.12.2.13 int Read2Bytes ()

Read the next two bytes from the current input stream into an int, and return that int. The bytes of the int, from lowest to highest, will correspond to the bytes read from the stream, from last to first. The highest two bytes will be 0.

Each byte read will be written to all registered capture buffers.

Returns

an int representing the 2 bytes read, as described above.

Exceptions

Asn1EndOfBufferException if at end-of-stream

3.12.2.14 int Read4Bytes ()

Read the next four bytes from the current input stream into an int, and return that int. The bytes of the int, from lowest to highest, will correspond to the bytes read from the stream, from last to first.

Each byte read will be written to all registered capture buffers.

Returns

an int representing the 4 bytes read, as described above.

Exceptions

Asn1EndOfBufferException if at end-of-stream

3.12.2.15 **abstract int ReadByte () [pure virtual]**

This abstract method returns the next available 8-bit value from the input stream. It is implemented differently for BER/DER and PER to take into account odd alignments in PER.

Returns

Next 8-bit byte value from input stream

3.12.2.16 **virtual void RemoveCaptureBuffer (System.IO.MemoryStream *buffer*) [virtual]**

This method is used to remove a capture buffer from the internal capture buffer list. The add and remove methods can be used to get a set of raw bytes from the input stream for further processing.

Parameters

buffer Buffer in which captured bytes stored

3.12.2.17 **virtual void Reset () [virtual]**

This method is used to reset the current position in the decode buffer back to the location of the last 'mark' call.

3.12.2.18 **virtual void SetInputStream (byte[] *msgdata*, int *offset*, int *length*) [virtual]**

This method will set the input stream from which data is read. This version of the method allows a byte array containing encoded data to be specified.

Parameters

msgdata Byte array containing encoded message data

offset Starting offset of data in the byte array

length Length (in bytes) of the encoded data

3.12.2.19 **virtual long Skip (long *nbytes*) [virtual]**

This method will skip over the requested number of bytes in the input stream.

Parameters

nbytes Number of bytes to skip

Returns

Skipped number of bytes

3.12.3 **Member Data Documentation**

3.12.3.1 **internal int mByteCount [protected]**

This member variable holds the count of bytes currently read from the message being decoded or input stream.

3.12.4 Property Documentation

3.12.4.1 virtual int ByteCount [get]

Gets the count of bytes currently read from the message being decoded or input stream.

3.12.4.2 bool LazyOpenTypeDecode [get, set]

Lazy open type decoding. This property is relevant only when generating table constraint code (otherwise, open types cannot be decoded). Generated decode methods check this property to determine whether to decode open types or not. When lazy open type decoding is turned on, you can use the generated decodeOpenType* methods to decode open types (again, assuming table constraint code was generated).

3.13 Asn1DiscreteCharSet Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1CharSet](#).

Public Member Functions

- [Asn1DiscreteCharSet](#) (int[] charSet)
- [Asn1DiscreteCharSet](#) (System.String charSet)
- override int [GetCharAtIndex](#) (int index)
- override int [GetCharIndex](#) (int charValue)
- override bool [validate](#) (String s)

Protected Member Functions

- bool [helpValidate](#) (char c)

Properties

- override int [MaxValue](#) [get]

3.13.1 Detailed Description

This class is used to represent a discrete set of characters from a permitted alphabet.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 Asn1DiscreteCharSet (System.String *charSet*)

This constructor sets the permitted alphabet character set

Parameters

charSet Permitted alphabet character set

3.13.2.2 Asn1DiscreteCharSet (int[] *charSet*)

This constructor sets the permitted alphabet character set

Parameters

charSet Permitted alphabet character set

3.13.3 Member Function Documentation

3.13.3.1 override int GetCharAtIndex (int *index*) [virtual]

This method will fetch the character from the permitted alphabet at the given index.

Parameters

index Index of character within the character set

Returns

Character at given index

Exceptions

[*Asn1ConsVioException*](#) Thrown if index not within define range

Implements [Asn1CharSet](#).

3.13.3.2 `override int GetCharIndex (int charValue) [virtual]`

This method will determine the index of the given character within the permitted alphabet character set.

Parameters

charValue Character value to search for

Returns

Index of character

Exceptions

[*Asn1ConsVioException*](#) thrown if '*charValue*' not found in set

Implements [Asn1CharSet](#).

3.13.3.3 `bool helpValidate (char c) [protected]`

This function helps validate a character string by checking a character to see if it is in the character set. It returns false if a character is not contained in the set and true otherwise.

3.13.3.4 `override bool validate (String s) [virtual]`

This method will validate a character string by comparing its contents to the character set. If a character string contains characters that are not in the character set, this method will return false. Otherwise it returns true.

Parameters

s The string to be validated.

Returns

False if the string contains invalid characters; true otherwise.

Implements [Asn1CharSet](#).

3.13.4 Property Documentation

3.13.4.1 `override int MaxValue` [get]

Gets Upper Bound Character or Character with max int value. It will determine the maximum value of the given character within the permitted alphabet character set. As the charset is canonical order, max value is of the last character.

Reimplemented from [Asn1CharSet](#).

3.14 Asn1EncodeBuffer Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1MessageBuffer](#).

Public Member Functions

- abstract void [BinDump](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [BinDump](#) (System.String varName)
- abstract void [Copy](#) (byte[] value, int offset, int len)
- void [Copy](#) (byte[] value)
- abstract void [Copy](#) (byte value)
- override Stream [GetInputStream](#) ()
- Stream [GetOutputStream](#) ()
- virtual void [HexDump](#) (System.IO.StreamWriter outs)
- virtual void [HexDump](#) ()
- abstract void [Reset](#) ()
- abstract void [Write](#) (System.IO.Stream outs)

Static Public Member Functions

- static int [EncodeIntSigned](#) (long value, byte[] dest, int offset)
- static int [EncodeIntUnsigned](#) (long value, byte[] dest, int offset)
- static int [GetMinimalOctetsSigned](#) (long value)
- static int [GetMinimalOctetsUnsigned](#) (long value)

Public Attributes

- const int [SIZE_INCREMENT](#) = 1024

Protected Attributes

- internal int [mSizeIncrement](#)

Properties

- abstract System.IO.MemoryStream [ByteArrayInputStream](#) [get]
- abstract byte[] [MsgCopy](#) [get]
- abstract int [MsgLength](#) [get]

3.14.1 Detailed Description

This is the base class to specific encode buffer classes for the different types of encoding rules (BER, DER and PER).

3.14.2 Member Function Documentation

3.14.2.1 abstract void BinDump (System.IO.StreamWriter *outs*, System.String *varName*) [pure virtual]

This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.

Parameters

outs Output will be written to this stream

varName Name of the Decoded ASN1 Type

3.14.2.2 virtual void BinDump (System.String *varName*) [virtual]

This method invokes an overloaded version of BinDump to dump the encoded message to standard output.

Parameters

varName Name of the Decoded ASN1 Type

3.14.2.3 abstract void Copy (byte[] *value*, int *offset*, int *len*) [pure virtual]

This method copies multiple bytes from the given array to the encode buffer.

Parameters

value Array of bytes to copy to the encode buffer

offset The first byte copied is value[offset]

len The last byte copied is value[offset + len - 1]

3.14.2.4 void Copy (byte[] *value*)

This method copies multiple bytes to the encode buffer

Parameters

value Array of bytes to copy to the encode buffer

3.14.2.5 abstract void Copy (byte *value*) [pure virtual]

This abstract method is used to copy a single byte to the encode buffer.

Parameters

value The byte value to copy

3.14.2.6 `static int EncodeIntSigned (long value, byte[] dest, int offset) [static]`

Encode an integer value as a signed value (2's complement), in the minimum number of octets possible (≥ 1), into the given array. /p>

Parameters

value The value to encode.

dest The array to but the encoding into. It must be large enough to hold the result, taking into account the given offset.

offset The index where the first byte should go.

Returns

The number of bytes.

3.14.2.7 `static int EncodeIntUnsigned (long value, byte[] dest, int offset) [static]`

Encode an integer value as an unsigned value (binary integer), in the minimum number of octets possible (≥ 1), into the given array.

Parameters

value The value to encode. It must be non-negative.

dest The array to but the encoding into. It must be large enough to hold the result, taking into account the given offset.

offset The index where the first byte should go.

Returns

The number of bytes.

3.14.2.8 `override Stream GetInputStream () [virtual]`

This method returns an input stream representing the encoded message.

Returns

Input stream containing encoded message

Implements [Asn1MessageBuffer](#).

3.14.2.9 `static int GetMinimalOctetsSigned (long value) [static]`

Return the minimal number of octets required to represent the given value, when a signed representation is being used.

Returns

The number of octets, ≤ 8 .

3.14.2.10 **static int GetMinimalOctetsUnsigned (long *value*) [static]**

Return the minimal number of octets required to represent the given value, when an unsigned representation is being used.

Parameters

value Must be ≥ 0

Returns

The number of octets, ≤ 8 .

3.14.2.11 **Stream GetOutputStream ()**

Return an output stream associated with this object. Anything written to the Stream will be written to this buffer using `copy(byte)` or `copy(byte[])`.

3.14.2.12 **virtual void HexDump (System.IO.StreamWriter *outs*) [virtual]**

This method dumps the encoded message in hex/ascii format to the given print output stream.

Parameters

outs Output stream object reference

3.14.2.13 **virtual void HexDump () [virtual]**

This method dumps the encoded message in hex/ascii format to the standard output stream.

3.14.2.14 **abstract void Reset () [pure virtual]**

This method resets the buffer to allow a new record to be encoded into it. Any previously encoded data is lost.

3.14.2.15 **abstract void Write (System.IO.Stream *outs*) [pure virtual]**

This method writes the encoded record to the given output stream.

Parameters

outs Output stream to which record is to be written

3.14.3 **Member Data Documentation**

3.14.3.1 **internal int mSizeIncrement [protected]**

This variable holds the user defined buffer increment size. It defines initial size and size it will be incremented by each time the buffer expands.

3.14.3.2 `const int SIZE_INCREMENT = 1024`

This constant specifies the default size of the encode buffer and size it will be incremented by each time the buffer expands. It is currently set to 1024 bytes.

3.14.4 Property Documentation

3.14.4.1 `abstract System.IO.MemoryStream ByteArrayInputStream [get]`

Gets a reference to a `MemoryStream` representing the encoded message. This is the preferred way to access the contents of the encoded message as it is the most efficient (it does not make a copy of the message).

Value: `MemoryStream` containing encoded message

3.14.4.2 `abstract byte [] MsgCopy [get]`

Gets the encoded message in a byte array. This is less efficient than the `GetInputStream` method because the message contents must be copied to a newly created byte array.

Value: byte array containing encoded message

3.14.4.3 `abstract int MsgLength [get]`

Gets the length (in bytes) of the encoded message component.

Value: length of encoded message component

3.15 Asn1EndOfBufferException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1EndOfBufferException](#) ([Asn1DecodeBuffer](#) buffer)

3.15.1 Detailed Description

This class defines the 'ASN.1 end of buffer' exception that is thrown when an unexpected end-of-buffer condition is encountered when decoding a message..

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Asn1EndOfBufferException (Asn1DecodeBuffer *buffer*)

This constructor creates an exception object with a textual message describing the tag of the duplicate element..

Parameters

buffer Decode buffer object reference

3.16 Asn1Enumerated Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1Enumerated](#) (int value_)
- [Asn1Enumerated](#) ()
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- virtual void [Encode](#) (Asn1XmlEncoder buffer, String elemName, String nsPrefix, bool asText)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XerEncodeBuffer buffer)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (int value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

Static Public Member Functions

- static int [ParseValue](#) (System.String value)

Public Attributes

- int [mValue](#)
- const int [UNDEFINED](#) = - 999

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Properties

- int [Value](#) [get]

3.16.1 Detailed Description

This class represents the ASN.1 ENUMERATED built-in type. It is declared to be an abstract class and therefore cannot be used on its own. It must be extended by a specific enumerated type class.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 Asn1Enumerated ()

The default constructor sets the enumerated value to undefined.

3.16.2.2 Asn1Enumerated (int value_)

This constructor creates an enumerated object from a integer value.

Parameters

value_ Integer value

3.16.3 Member Function Documentation

3.16.3.1 virtual void Encode (Asn1PerOutputStream outs, long lower, long upper) [virtual]

This method encodes an ASN.1 enumerated value using the Packed Encoding Rules (PER).

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Output Stream object

lower Smallest enumerated value in the set

upper Largest enumerated value in the set

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.16.3.2 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 enumerated value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.16.3.3 virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode the enumerated value to JSON.

Parameters

outstream

3.16.3.4 virtual void Encode (Asn1XmlEncoder *buffer*, String *elemName*, String *nsPrefix*, bool *asText*) [virtual]

This method encodes an ASN.1 enumerated value. It is for use with extended-XER.

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix XML element name space prefix

asText If true, encode the value as XML text, otherwise encode as an empty element.

3.16.3.5 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an ASN.1 enumerated value using the Obj-Sys XML Encoding rules. It encodes the value as XML text.

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

3.16.3.6 virtual void Encode (Asn1XerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 enumerated value using the XML encoding rules (XER). This method does not add start and end tags (<tag> and </tag>), only value is encoded (<val1/> or <val2/>).

Parameters

buffer Encode message buffer object

3.16.3.7 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes an ASN.1 enumerated value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.16.3.8 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

Encode enumerated value according to OER.

Reimplemented from [Asn1Type](#).

3.16.3.9 virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This method encodes an ASN.1 enumerated value using the Packed Encoding Rules (PER).

Parameters

buffer PER Encode message buffer object

lower Smallest enumerated value in the set

upper Largest enumerated value in the set

3.16.3.10 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 enumerated value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.16.3.11 override bool Equals (System.Object *value*)

This method compares this enumerated value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1Enumerated](#).

Returns

true if the specified Object is equal to the current Object; otherwise, false.

3.16.3.12 virtual bool Equals (int *value*) [virtual]

This method compares this enumerated value to the given value for equality.

Parameters

value The int or enumerated value to compare with the current Object.

Returns

true if the specified int value is equal to the current Object; otherwise, false.

3.16.3.13 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.16.3.14 **static int ParseValue (System.String value) [static]**

This method will parse the given enumeration text and set the enumerated value. This method is implemented by the extending class for XER or XML code generation ONLY.

Parameters

value enumeration text

Returns

Stringified representation of the value

AB: don't make it abstract (it is only for XER)

3.16.3.15 **override System.String ToString ()**

This method will return the enumeration text for a given enumerated value.

Returns

Stringified representation of the value

3.16.4 **Member Data Documentation**

3.16.4.1 **new readonly Asn1Tag _TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 10).

Reimplemented from [Asn1Type](#).

3.16.4.2 **int mValue**

This public member variable is where the enumerated value is stored. This is the value that is encoded when one of the encode methods is called. It is also where the decoded result is stored when a Decode method is called.

3.16.4.3 **const int UNDEFINED = - 999**

The `UNDEFINED` constant is stored in the `mValue` member variable when the value of this enumerated type is undetermined.

3.16.5 Property Documentation

3.16.5.1 int Value [get]

The integer value associated with this enumerated value.

3.17 Asn1Exception Class Reference

Inherited by [Asn1ConsVioException](#), [Asn1EndOfBufferException](#), [Asn1InvalidArgException](#), [Asn1InvalidChoiceOptionException](#), [Asn1InvalidEnumException](#), [Asn1InvalidLengthException](#), [Asn1InvalidObjectIDException](#), [Asn1MissingRequiredException](#), [Asn1SeqOrderException](#), and [Asn1ValueParseException](#).

Public Member Functions

- [Asn1Exception](#) ([Asn1DecodeBuffer](#) buffer, System.String message)
- [Asn1Exception](#) (System.String message, System.Exception innerException)
- [Asn1Exception](#) (System.String message)

3.17.1 Detailed Description

This class defines a generic ASN.1 exception for use as a base class for exceptions common to all encode/decode operations. Specific exceptions for BER, DER, and PER encoding and decoding are subclassed from this base class..

3.17.2 Constructor & Destructor Documentation

3.17.2.1 Asn1Exception (System.String message)

This constructor passes the given message text to the superclass.

Parameters

message Error message text

3.17.2.2 Asn1Exception (System.String message, System.Exception innerException)

This constructor passes the given message text to the superclass.

Parameters

message Error message text

innerException The exception that is the cause of the current exception.

3.17.2.3 Asn1Exception (Asn1DecodeBuffer buffer, System.String message)

This constructor creates the base exception object and captures the current buffer offset from the Decode buffer..

Parameters

buffer ASN.1 Decode buffer object reference

message Error message text

3.18 Asn1GeneralizedTime Class Reference

Public Member Functions

- [Asn1GeneralizedTime](#) (System.String data, bool useDerRules)
- [Asn1GeneralizedTime](#) (System.String data)
- [Asn1GeneralizedTime](#) (bool useDerRules)
- [Asn1GeneralizedTime](#) ()
- override System.Int32 [CompareTo](#) (System.Object obj)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [ParseString](#) (System.String data)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Protected Member Functions

- internal override bool [CompileString](#) ()

Properties

- virtual int [Century](#) [get, set]

3.18.1 Detailed Description

This is a container class for holding the components of an ASN.1 generalized time string value.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 [Asn1GeneralizedTime](#) ()

The default constructor creates an empty time string object.

3.18.2.2 [Asn1GeneralizedTime](#) (bool *useDerRules*)

This constructor creates an empty time string object and allows DER encoding rules to be specified.

Parameters

useDerRules 'true' if time string should be encoded with DER/PER.

3.18.2.3 `Asn1GeneralizedTime` (`System.String data`)

This version of the constructor can be used to set the string `mValue` member variable to the given time string. The format of a `GeneralizedTime` string is `YYYYMMDDHHMMSS.n[Z][-HHMM]`.

Parameters

data Character string

3.18.2.4 `Asn1GeneralizedTime` (`System.String data`, `bool useDerRules`)

This version of the constructor can be used to set the string `mValue` member variable to the given time string and specify DER encoding rules be used to construct the string.

Parameters

data Character string

useDerRules 'true' if time string should be encoded with DER/PER.

3.18.3 Member Function Documentation

3.18.3.1 `override System.Int32 CompareTo` (`System.Object obj`)

This method compares this object with `Asn1Time` class instance or with `System.DateTime` instance. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. Note that the action of this method may differentiate for different inherited `Asn1Time` classes.

Parameters

obj the Object to be compared.

Returns

The difference in Ticks with the specified object.

3.18.3.2 `internal override bool CompileString` () [`protected`]

Compiles new time string accoring X.680 (clause 41) and ISO 8601.

Returns

true, if succeed, or false code, if error.

3.18.3.3 `override void Decode` (`Asn1BerDecodeBuffer buffer`, `bool explicitTagging`, `int implicitLength`)

This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

3.18.3.4 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*)

This method encodes and writes to stream an ASN.1 generalized time string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

3.18.3.5 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*)

This method encodes an ASN.1 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

3.18.3.6 override void ParseString (System.String *data*)

This method parses the given time string value. It will throw an exception if the string is not in the valid time format. The valid format of a GeneralizedTime string is YYYYMMDDHHMMSS.n[Z][*-HHMM*].

Parameters

data The time string value to be parsed.

Exceptions

Asn1Exception Thrown, if operation is failed.

3.18.4 Member Data Documentation

3.18.4.1 new readonly Asn1Tag _TAG [static]

The *_TAG* constant describes the universal tag for this data type (UNIVERSAL 24).

3.18.5 Property Documentation

3.18.5.1 virtual int Century [get, set]

Gets or Sets the century part (first two digits) of the year component of the time value.

Value: Century part (first two digits) of the year component.

Exceptions

Asn1Exception Thrown, if operation is failed.

3.19 Asn1GeneralString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString](#).

Public Member Functions

- [Asn1GeneralString](#) (System.String data)
- [Asn1GeneralString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.19.1 Detailed Description

This is a container class for holding the components of an ASN.1 general string value.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 Asn1GeneralString ()

The default constructor creates an empty string object.

3.19.2.2 Asn1GeneralString (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string value.

Parameters

data Character string

3.19.3 Member Function Documentation

3.19.3.1 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.19.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 general string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.19.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.19.4 Member Data Documentation

3.19.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 27).

Reimplemented from [Asn1Type](#).

3.20 Asn1GraphicString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString](#).

Public Member Functions

- [Asn1GraphicString](#) (System.String data)
- [Asn1GraphicString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.20.1 Detailed Description

This is a container class for holding the components of an ASN.1 graphic string value.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 Asn1GraphicString ()

The default constructor creates an empty string object.

3.20.2.2 Asn1GraphicString (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string value.

Parameters

data string representation of GraphicString

3.20.3 Member Function Documentation

3.20.3.1 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 graphic string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.20.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 graphic string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.20.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 graphic string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.20.4 Member Data Documentation

3.20.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 25).

Reimplemented from [Asn1Type](#).

3.21 Asn1IA5String Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn18BitCharString](#).

Public Member Functions

- [Asn1IA5String](#) (System.String data)
- [Asn1IA5String](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.21.1 Detailed Description

This is a container class for holding the components of an ASN.1 IA5 string value.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 [Asn1IA5String](#) ()

The default constructor creates an empty string object.

3.21.2.2 [Asn1IA5String](#) (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data string representation of IA5String

3.21.3 Member Function Documentation

3.21.3.1 override void [Decode](#) (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 IA5 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.21.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 IA5 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.21.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 IA5 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.21.4 Member Data Documentation

3.21.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 22).

Reimplemented from [Asn1Type](#).

3.22 Asn1InputStream Interface Reference

Public Member Functions

- int [Available](#) ()
- void [Close](#) ()
- void [Mark](#) ()
- bool [MarkSupported](#) ()
- void [Reset](#) ()
- long [Skip](#) (long nbytes)

3.22.1 Detailed Description

This interface is a base interface for all classes, which implement an input stream functionality for decoding.

3.22.2 Member Function Documentation

3.22.2.1 int Available ()

Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream. The next caller might be the same thread or or another thread.

Returns

the number of bytes that can be read from this input stream without blocking.

Exceptions

System.SystemException if an I/O error occurs.

3.22.2.2 void Close ()

Closes this input stream and releases any system resources associated with the stream.

Exceptions

System.SystemException if an I/O error occurs.

3.22.2.3 void Mark ()

This method is used to mark the current position in the input stream for retry processing or resetting the input stream position to current position.

3.22.2.4 bool MarkSupported ()

Tests if this input stream supports the seeking. This method is equivalent to C# `CanSeek` method of `System.IO.Stream`.

Returns

`true` if input stream supports seeking; Otherwise `false`.

3.22.2.5 void Reset ()

This method is used to reset the current position in the input stream back to the location of the last 'mark' call. It is equivalent to calling 'Stream.Position' to marked location.

3.22.2.6 long Skip (long *nbytes*)

This method will skip over the requested number of bytes in the input stream.

Parameters

nbytes Number of bytes to skip

Exceptions

System.SystemException if an I/O error occurs.

Returns

Skipped number of bytes

3.23 Asn1Integer Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1Integer](#) (long value)
- [Asn1Integer](#) ()
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, Object lower, Object upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, Object upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, Object lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [Decode16Bit](#) (Asn1MderDecodeBuffer buffer, bool signed)
- void [Decode32Bit](#) (Asn1MderDecodeBuffer buffer, bool signed)
- void [Decode8Bit](#) (Asn1MderDecodeBuffer buffer, bool signed)
- void [DecodeSigned](#) (Asn1OerDecodeBuffer buffer)
- void [DecodeSigned](#) (Asn1OerDecodeBuffer buffer, int octets)
- void [DecodeUnsigned](#) (Asn1OerDecodeBuffer buffer)
- void [DecodeUnsigned](#) (Asn1OerDecodeBuffer buffer, int octets)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, Object lower, Object upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, Object upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, Object lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outs)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, Object lower, Object upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, Object upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, Object lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- void [Encode16Bit](#) (Asn1MderOutputStream outs, bool signed)
- void [Encode32Bit](#) (Asn1MderOutputStream outs, bool signed)
- void [Encode8Bit](#) (Asn1MderOutputStream outs, bool signed)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- void [EncodeSigned](#) (Asn1OerEncodeBuffer buffer, int octets)
- void [EncodeSigned](#) (Asn1OerEncodeBuffer buffer)
- void [EncodeUnsigned](#) (Asn1OerEncodeBuffer buffer, int octets)
- void [EncodeUnsigned](#) (Asn1OerEncodeBuffer buffer)
- override bool [Equals](#) (System.Object value)

- virtual bool [Equals](#) (long value)
- virtual int [GetBitCount](#) ()
- override int [GetHashCode](#) ()
- virtual int [GetUnsignedBitCount](#) ()
- override System.String [ToString](#) ()

Static Public Member Functions

- static long [DecodeValue](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- static long [DecodeValue](#) (Asn1PerDecodeBuffer buffer)
- static void [EncodeValue](#) (Asn1PerEncoder encoder, long val, long lower, long upper)
- static int [GetBitCount](#) (long ivalue)
- static int [GetUnsignedBitCount](#) (long ivalue)

Public Attributes

- long [mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.23.1 Detailed Description

This class represents the ASN.1 INTEGER built-in type.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 [Asn1Integer](#) ()

The default constructor sets the integer value to zero.

3.23.2.2 [Asn1Integer](#) (long *value*)

This constructor creates an integer object from a integer value.

Parameters

value Integer value

3.23.3 Member Function Documentation

3.23.3.1 virtual void [Decode](#) (Asn1JsonDecodeBuffer *buffer*) [[virtual](#)]

Decode ASN.1 INTEGER from JSON.

3.23.3.2 **override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]**

Decode this value as if unconstrained. Subclasses may override this method to decode according to the constraints on the respective ASN.1 type.

Reimplemented from [Asn1Type](#).

3.23.3.3 **virtual void Decode (Asn1PerDecodeBuffer *buffer*, Object *lower*, Object *upper*) [virtual]**

This method decodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The decoded result is stored in the public member 'mValue' in this object.

Parameters

buffer PER Decode message buffer object

lower Lower bound equal MIN

upper Upper bound equal MAX

3.23.3.4 **virtual void Decode (Asn1PerDecodeBuffer *buffer*, long *lower*, Object *upper*) [virtual]**

This method decodes a semi-constrained ASN.1 integer value using the Packed Encoding Rules (PER). The decoded result is stored in the public member 'mValue' in this object.

Parameters

buffer PER Decode message buffer object

lower Lower bound of the integer range

upper Upper bound equal MAX

3.23.3.5 **virtual void Decode (Asn1PerDecodeBuffer *buffer*, Object *lower*, long *upper*) [virtual]**

This method decodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The decoded result is stored in the public member 'mValue' in this object.

Parameters

buffer PER Decode message buffer object

lower Lower bound equal MIN

upper Upper bound of the integer range

3.23.3.6 **virtual void Decode (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [virtual]**

This method decodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER). The decoded result is stored in the public member 'mValue' in this object.

Parameters

buffer PER Decode message buffer object

lower Lower bound of the integer range

upper Upper bound of the integer range

3.23.3.7 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are decoded. The decoded result is stored in the public member 'mValue' in this object.

Parameters

buffer PER Decode message buffer object

Reimplemented from [Asn1Type](#).

3.23.3.8 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.23.3.9 void Decode16Bit (Asn1MderDecodeBuffer *buffer*, bool *signed*)

Decode a signed or unsigned 16-bit integer from an MDER encoding. This should be used to decode integer types that are constrained to exactly the value space of a signed/unsigned 16 bit integer.

3.23.3.10 void Decode32Bit (Asn1MderDecodeBuffer *buffer*, bool *signed*)

Decode a signed or unsigned 32-bit integer from an MDER encoding. This should be used to decode integer types that are constrained to exactly the value space of a signed/unsigned 32 bit integer.

3.23.3.11 void Decode8Bit (Asn1MderDecodeBuffer *buffer*, bool *signed*)

Decode a signed or unsigned 8-bit integer from an MDER encoding. This should be used to decode integer types that are constrained to exactly the value space of a signed/unsigned 8 bit integer.

3.23.3.12 void DecodeSigned (Asn1OerDecodeBuffer *buffer*)

Decode a variable-size signed integer, encoded with a length, into this object, according to OER.

3.23.3.13 void DecodeSigned (Asn1OerDecodeBuffer *buffer*, int *octets*)

Decode a signed integer (2's complement) of the given length into this object.

Parameters

octets The number of octets in which the value was encoded. $0 < \text{octets} \leq 8$

3.23.3.14 void DecodeUnsigned (Asn1OerDecodeBuffer *buffer*)

Decode a variable-size unsigned integer, encoded with a length, into this object, according to OER.

3.23.3.15 void DecodeUnsigned (Asn1OerDecodeBuffer *buffer*, int *octets*)

Decode an unsigned integer (binary integer) of the given length into this object.

Parameters

octets The number of octets in which the value was encoded. $0 < \text{octets} \leq 8$

3.23.3.16 static long DecodeValue (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [static]

This method decodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER). The decoded result is returned.

Parameters

buffer PER Decode message buffer object

lower Lower bound of the integer range

upper Upper bound of the integer range

3.23.3.17 static long DecodeValue (Asn1PerDecodeBuffer *buffer*) [static]

This method decodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are decoded. The decoded result is returned.

Parameters

buffer PER Decode message buffer object

3.23.3.18 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 integer value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.23.3.19 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 integer value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.23.3.20 virtual void Encode (Asn1PerOutputStream *outs*, Object *lower*, Object *upper*) [virtual]

This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

outs PER Encode message buffer object

lower Lower bound equal MIN

upper Upper bound equal MAX

3.23.3.21 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, Object *upper*) [virtual]

This method encodes a semi-constrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

outs PER Encode message buffer object

lower Lower bound (inclusive) of integer being encoded

upper Upper bound equal MAX

3.23.3.22 virtual void Encode (Asn1PerOutputStream *outs*, Object *lower*, long *upper*) [virtual]

This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

outs PER Encode message buffer object

lower Lower bound equal MIN

upper Upper bound (inclusive) of integer being encoded

3.23.3.23 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, long *upper*) [virtual]

This method encodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Encode message buffer object

lower Lower bound (inclusive) of integer being encoded

upper Upper bound (inclusive) of integer being encoded

Exceptions

[*Asn1Exception*](#) Thrown, if operation is failed.

3.23.3.24 **override void Encode (Asn1PerOutputStream outs) [virtual]**

This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Encode message buffer object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.23.3.25 **override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]**

This method encodes and writes to the stream an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.23.3.26 **virtual void Encode (Asn1JsonOutputStream outs) [virtual]**

Encode the value of this object as a JSON number.

Parameters

out

3.23.3.27 **override void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix) [virtual]**

This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

3.23.3.28 **override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]**

This method encodes an ASN.1 integer value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.23.3.29 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

Encode this value as if unconstrained. Subclasses may override this method to encode it according to the constraints on the respective ASN.1 type.

Reimplemented from [Asn1Type](#).

3.23.3.30 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, Object *lower*, Object *upper*) [virtual]**

This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

lower Lower bound equal MIN

upper Upper bound equal MAX

3.23.3.31 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, Object *upper*) [virtual]**

This method encodes a semi-constrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

lower Lower bound (inclusive) of integer being encoded

upper Upper bound equal MAX

3.23.3.32 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, Object *lower*, long *upper*) [virtual]**

This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

lower Lower bound equal MIN

upper Upper bound (inclusive) of integer being encoded

3.23.3.33 virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This method encodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

- buffer* PER Encode message buffer object
- lower* Lower bound (inclusive) of integer being encoded
- upper* Upper bound (inclusive) of integer being encoded

3.23.3.34 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

- buffer* PER Encode message buffer object

Reimplemented from [Asn1Type](#).

3.23.3.35 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

- buffer* Encode message buffer object
- explicitTagging* Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.23.3.36 void Encode16Bit (Asn1MderOutputStream *outs*, bool *signed*)

This method encodes this ASN.1 INTEGER value to the MDER encoding. The value must fall in the set of 16-bit unsigned integers (if signed is false) or 16-bit signed integers (if signed is true).

3.23.3.37 void Encode32Bit (Asn1MderOutputStream *outs*, bool *signed*)

This method encodes this ASN.1 INTEGER value to the MDER encoding. The value must fall in the set of 32-bit unsigned integers (if signed is false) or 32-bit signed integers (if signed is true).

3.23.3.38 void Encode8Bit (Asn1MderOutputStream *outs*, bool *signed*)

This method encodes this ASN.1 INTEGER value to the MDER encoding. The value must fall in the set of 8-bit unsigned integers (if signed is false) or 8-bit signed integers (if signed is true).

3.23.3.39 **override void EncodeAttribute (Asn1XmlEncoder *buffer*, System.String *attrName*) [virtual]**

This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name

attribute Element attribute value

Reimplemented from [Asn1Type](#).

3.23.3.40 **void EncodeSigned (Asn1OerEncodeBuffer *buffer*, int *octets*)**

Encode integer value as a signed value (2's complement) in the given number of octets.

The value must fit in the given number of octets.

3.23.3.41 **void EncodeSigned (Asn1OerEncodeBuffer *buffer*)**

Encode this value as a variable-size signed integer, with length, according to OER.

3.23.3.42 **void EncodeUnsigned (Asn1OerEncodeBuffer *buffer*, int *octets*)**

Encode integer value as an unsigned value (binary integer) in the given number of octets, according to OER.

The value must be non-negative and fit in the given number of octets.

Parameters

buffer The buffer

octets The number of octets to encode in: 1, 2, 4, or 8.

3.23.3.43 **void EncodeUnsigned (Asn1OerEncodeBuffer *buffer*)**

Encode this value as a variable-size unsigned integer, with length, according to OER.

3.23.3.44 **static void EncodeValue (Asn1PerEncoder *encoder*, long *val*, long *lower*, long *upper*) [static]**

This method encodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

lower Lower bound (inclusive) of integer being encoded

upper Upper bound (inclusive) of integer being encoded

3.23.3.45 **override bool Equals (System.Object value)**

This method compares this integer value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1Integer](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.23.3.46 **virtual bool Equals (long value) [virtual]**

This method compares this integer value to the given value for equality.

Parameters

value The long value to compare with the current Object.

Returns

`true` if the specified long value is equal to the current Object; otherwise, `false`.

3.23.3.47 **virtual int GetBitCount () [virtual]**

This method calculates the count of bits in the contained integer value.

Returns

Bit count.

3.23.3.48 **static int GetBitCount (long ivalue) [static]**

This method calculates the count of bits in an integer value.

Parameters

ivalue Integer value in which to count bits.

Returns

Bit count.

3.23.3.49 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.23.3.50 **virtual int GetUnsignedBitCount () [virtual]**

This method calculates the count of bits in the contained unsigned integer value.

Returns

Bit count.

3.23.3.51 **static int GetUnsignedBitCount (long ivalue) [static]**

This method calculates the count of bits in an unsigned integer value.

Parameters

ivalue Integer value in which to count bits.

Returns

Bit count.

3.23.3.52 **override System.String ToString ()**

This method will return a string representation of the integer value. The format is the ASN.1 value format for this type.

Returns

Stringified representation of the value

3.23.4 **Member Data Documentation**

3.23.4.1 **new readonly Asn1Tag _TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 2).

Reimplemented from [Asn1Type](#).

3.23.4.2 **long mValue**

This public member variable is where the integer value is stored. This is the value that is encoded when one of the encode methods is called. It is also where the decoded result is stored when a Decode method is called.

3.24 Asn1InvalidArgException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1InvalidArgException](#) (System.String argName, System.String argValue)

3.24.1 Detailed Description

This class defines the 'ASN.1 invalid argument' exception that is thrown when an argument that is passed to a method is determined to be invalid (for example, not within a defined range)..

3.24.2 Constructor & Destructor Documentation

3.24.2.1 Asn1InvalidArgException (System.String *argName*, System.String *argValue*)

This constructor creates an exception object with a textual message describing the argument that was invalid..

Parameters

argName Name of invalid argument

argValue Value of invalid argument

3.25 Asn1InvalidChoiceOptionException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1InvalidChoiceOptionException](#) ()
- [Asn1InvalidChoiceOptionException](#) (Asn1PerDecodeBuffer buffer, int index)
- [Asn1InvalidChoiceOptionException](#) (Asn1BerDecodeBuffer buffer, [Asn1Tag](#) tag)

3.25.1 Detailed Description

This class defines the 'ASN.1 invalid choice option' exception that is thrown when a CHOICE construct is detected to contain an element that is not within the given set.

3.25.2 Constructor & Destructor Documentation

3.25.2.1 [Asn1InvalidChoiceOptionException](#) (Asn1BerDecodeBuffer *buffer*, [Asn1Tag](#) *tag*)

This constructor creates an exception object with a textual message describing the tag of the invalid element..

Parameters

buffer BER Decode buffer object reference

tag Tag value of duplicate element

3.25.2.2 [Asn1InvalidChoiceOptionException](#) (Asn1PerDecodeBuffer *buffer*, int *index*)

This constructor creates an exception object with a textual message describing the PER choice index of the invalid element..

Parameters

buffer PER Decode buffer object reference

index Parsed choice index value

3.25.2.3 [Asn1InvalidChoiceOptionException](#) ()

The default constructor is invoked in the encode logic if the object assigned to the choice item is not in the allowed set..

3.26 Asn1InvalidEnumException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1InvalidEnumException](#) (System.String data)
- [Asn1InvalidEnumException](#) (long data)

3.26.1 Detailed Description

This class defines the 'ASN.1 invalid enum' exception that is thrown when an enumerated value is not within the defined set of values.

3.26.2 Constructor & Destructor Documentation

3.26.2.1 Asn1InvalidEnumException (long data)

This constructor creates an exception object with a default textual message with integer value.

Parameters

data Invalid enumerated value

3.26.2.2 Asn1InvalidEnumException (System.String data)

This constructor creates an exception object with a default textual message with textual enumerated value.

Parameters

data Invalid enumerated value

3.27 Asn1InvalidLengthException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1InvalidLengthException \(\)](#)

3.27.1 Detailed Description

This class defines the 'ASN.1 invalid length' exception that is thrown when a length is determined to be invalid.

Things that can cause this to be thrown are:

- Constructor length field is not sum of parts.
- Object identifier length is not sum of sub ID lengths.

3.27.2 Constructor & Destructor Documentation

3.27.2.1 Asn1InvalidLengthException ()

This constructor creates an exception object with a default textual message.

3.28 Asn1InvalidObjectIDException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1InvalidObjectIDException \(\)](#)

3.28.1 Detailed Description

This class defines the 'ASN.1 invalid object identifier' exception that is thrown when an Object Identifier is determined to be invalid.

Things that can cause this to be thrown are:

- Max subID's (128) exceeded.
- Not enough subID's in the OID (must contain at least 2).
- First subID > 2 and/or second subID > 40.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 Asn1InvalidObjectIDException ()

This constructor creates an exception object with a default textual message.

3.29 Asn1MessageBuffer Class Reference

Inherited by [Asn1DecodeBuffer](#), and [Asn1EncodeBuffer](#).

Public Member Functions

- virtual void [AddNamedEventHandler](#) ([Asn1NamedEventHandler](#) handler)
- abstract System.IO.Stream [GetInputStream](#) ()
- virtual void [InvokeCharacters](#) (System.String svalue)
- virtual void [InvokeEndElement](#) (System.String name, int index)
- virtual void [InvokeStartElement](#) (System.String name, int index)

Properties

- virtual [Asn1MessageBuffer](#) [EventHandlerList](#) [set]

3.29.1 Detailed Description

This is the base class for all of the different message buffer types. This includes the BER and PER encode and decode message buffer classes.

3.29.2 Member Function Documentation

3.29.2.1 virtual void AddNamedEventHandler (Asn1NamedEventHandler handler) [virtual]

This method adds a named event handler to the named event handler list for this buffer.

Parameters

handler [Asn1NamedEventHandler](#) object to be added

3.29.2.2 abstract System.IO.Stream GetInputStream () [pure virtual]

This abstract method must be implemented by all of the derived classes. It returns an input stream object reference to the message buffer contents (i.e. the encoded data).

Returns

Input stream object reference

Implemented in [Asn1DecodeBuffer](#), and [Asn1EncodeBuffer](#).

3.29.2.3 virtual void InvokeCharacters (System.String svalue) [virtual]

This method is used by the event handling logic to invoke the 'characters' event handling method when message contents are parsed. The `TypeCode` property is used for the event's type code.

Parameters

svalue Stringified representation of a parsed value field

3.29.2.4 virtual void InvokeEndElement (System.String *name*, int *index*) [virtual]

This method is used by the event handling logic to invoke the 'EndElement' event handling method when parsing of an element within a message is completed.

Parameters

name Name of the element

index Index of element if SEQUENCE OF or SET OF element

3.29.2.5 virtual void InvokeStartElement (System.String *name*, int *index*) [virtual]

This method is used by the event handling logic to invoke the 'StartElement' event handling method when parsing of an element within a message is started.

Parameters

name Name of the element

index Index of element if SEQUENCE OF or SET OF element

3.29.3 Property Documentation

3.29.3.1 virtual Asn1MessageBuffer EventHandlerList [set]

Sets the event dispatcher in this object to be equal to that in the given message buffer object. The two buffers will share the event dispatcher.

Value: Message buffer object

3.30 Asn1MissingRequiredException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1MissingRequiredException](#) (System.String elemName)
- [Asn1MissingRequiredException](#) (Asn1BerDecodeBuffer buffer)

3.30.1 Detailed Description

This class defines the 'ASN.1 set missing required element' exception that is thrown from Decode methods when a SET construct is decoded and found to be missing a required element..

3.30.2 Constructor & Destructor Documentation

3.30.2.1 Asn1MissingRequiredException (Asn1BerDecodeBuffer *buffer*)

This constructor creates an exception object with a textual message describing the error.

Parameters

buffer BER decode buffer object reference

3.30.2.2 Asn1MissingRequiredException (System.String *elemName*)

This constructor creates an exception object with a textual message describing the error including the name of the required element that is missing.

Parameters

elemName Name of missing required element

3.31 Asn1NamedEventHandler Interface Reference

Inherited by [Asn1TraceHandler](#).

Public Member Functions

- void [Characters](#) (System.String svalue, short typeCode)
- void [EndElement](#) (System.String name, int index)
- void [StartElement](#) (System.String name, int index)

3.31.1 Detailed Description

This interface defines the methods that must be implemented to define a SAX-like event handler. These methods are invoked from within the generated C# decode logic when significant events occur during the parsing of an ASN.1 message.

3.31.2 Member Function Documentation

3.31.2.1 void Characters (System.String svalue, short typeCode)

The Characters callback method is invoked when content (primitive data) is encountered. A stringified representation of the parsed value is returned.

Parameters

svalue Stringified representation of the parsed value. The representation will be in ASN.1 value format.

typeCode Identifier specifying the type of the parsed data variable. The enumerated list of values that might appear here is provided in the the [Asn1Type](#) class (see the documentation on this class for a full list of the names).

See also

<seealso cref=Asn1Type The type codes are member of [Asn1Type](#) class

Implemented in [Asn1TraceHandler](#).

3.31.2.2 void EndElement (System.String name, int index)

The EndElement callback method is invoked when the end of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is detected.

Parameters

name Name of the parsed element.

index Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implemented in [Asn1TraceHandler](#).

3.31.2.3 void StartElement (System.String *name*, int *index*)

The StartElement callback method is invoked when the start of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is encountered.

Parameters

name Name of the parsed element.

index Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implemented in [Asn1TraceHandler](#).

3.32 Asn1Null Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (object o)
- override System.String [ToString](#) ()

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)
- static readonly [Asn1Null NULL_VALUE](#) = new [Asn1Null](#)()

3.32.1 Detailed Description

This class represents the ASN.1 NULL built-in type.

3.32.2 Member Function Documentation

3.32.2.1 virtual void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 NULL from JSON.

3.32.2.2 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode ASN.1 NULL type, using Octet Encoding Rules (OER).

Reimplemented from [Asn1Type](#).

3.32.2.3 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.32.2.4 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 null value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.32.2.5 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 null value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.32.2.6 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 null value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.32.2.7 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.32.2.8 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 NULL value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.32.2.9 virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode this ASN.1 null value to JSON

3.32.2.10 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an ASN.1 null value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

3.32.2.11 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes an ASN.1 null value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.32.2.12 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

Encode ASN.1 NULL type, using Octet Encoding Rules (OER).

Reimplemented from [Asn1Type](#).

3.32.2.13 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.32.2.14 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 null value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version used the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length (in octets) of encoded component

Reimplemented from [Asn1Type](#).

3.32.2.15 **override bool Equals (object *o*)**

Tests for equality with any other object. Returns true if the input type is an [Asn1Null](#) object and false otherwise.

3.32.2.16 **override System.String ToString ()**

This method will return a string representation of the null value. The format is the ASN.1 value format for this type..

Returns

Stringified representation of the value

3.32.3 **Member Data Documentation**

3.32.3.1 **new readonly Asn1Tag _TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 5).

Reimplemented from [Asn1Type](#).

3.32.3.2 `readonly Asn1Null NULL_VALUE = new Asn1Null()` **[static]**

The `NULL_VALUE` constant represents a `NULL` value.

3.33 Asn1NumericString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn18BitCharString](#).

Public Member Functions

- [Asn1NumericString](#) (System.String data)
- [Asn1NumericString](#) ()
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.33.1 Detailed Description

This is a container class for holding the components of an ASN.1 numeric string value.

3.33.2 Constructor & Destructor Documentation

3.33.2.1 Asn1NumericString ()

The default constructor creates an empty string object.

3.33.2.2 Asn1NumericString (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string value.

Parameters

data string representation of numeric string

3.33.3 Member Function Documentation

3.33.3.1 virtual void Decode (Asn1PerDecodeBuffer buffer, long lower, long upper) [virtual]

This overloaded version of the Decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Decode message buffer object
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

3.33.3.2 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Decode message buffer object

Reimplemented from [Asn18BitCharString](#).

3.33.3.3 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

- buffer* Decode message buffer object
- explicitTagging* Flag indicating element is explicitly tagged
- implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.33.3.4 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 numeric string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

- outs* BER Output Stream object
- explicitTagging* Flag indicating explicit tagging should be done

Exceptions

- [Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.33.3.5 virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Encode message buffer object
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

3.33.3.6 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

- buffer* Encode message buffer object

Reimplemented from [Asn18BitCharString](#).

3.33.3.7 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

- buffer* Encode message buffer object
- explicitTagging* Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.33.4 Member Data Documentation

3.33.4.1 new readonly Asn1Tag _TAG [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 18).

Reimplemented from [Asn1Type](#).

3.34 Asn1ObjectDescriptor Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString](#).

Public Member Functions

- [Asn1ObjectDescriptor](#) (System.String data)
- [Asn1ObjectDescriptor](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.34.1 Detailed Description

This is a container class for holding the components of an ASN.1 object descriptor value.

3.34.2 Constructor & Destructor Documentation

3.34.2.1 Asn1ObjectDescriptor ()

The default constructor creates an empty string object.

3.34.2.2 Asn1ObjectDescriptor (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data string representation of ObjectDescriptor

3.34.3 Member Function Documentation

3.34.3.1 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.34.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 object descriptor value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.34.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.34.4 Member Data Documentation

3.34.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 7).

Reimplemented from [Asn1Type](#).

3.35 Asn1ObjectIdentifier Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Inherited by [Asn1RelativeOID](#).

Public Member Functions

- virtual void [Append](#) (int[] value2)
- [Asn1ObjectIdentifier](#) (int[] value)
- [Asn1ObjectIdentifier](#) ()
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (System.Object value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()
- String [ToXMLValue](#) ()

Public Attributes

- const int [MAXSUBIDS](#) = 128
- int[] [mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Protected Member Functions

- virtual void [Validate](#) ()

3.35.1 Detailed Description

This is a container class for holding the components of an ASN.1 object identifier value.

3.35.2 Constructor & Destructor Documentation

3.35.2.1 `Asn1ObjectIdentifier ()`

This constructor creates an empty object identifier that can be used in a Decode method call to receive an OID value.

3.35.2.2 `Asn1ObjectIdentifier (int[] value)`

This constructor initializes the object identifier from the given array of integer subidentifier values.

Parameters

value Array of subidentifiers

3.35.3 Member Function Documentation

3.35.3.1 `virtual void Append (int[] value2) [virtual]`

This method appends an object identifier value onto the existing value. A typical use of this method would be for SNMP objects to create the base and index parts.

Parameters

value2 Array of subidentifiers to append

3.35.3.2 `virtual void Decode (Asn1JsonDecodeBuffer buffer) [virtual]`

Decode ASN.1 OBJECT-IDENTIFIER from JSON.

3.35.3.3 `override void Decode (Asn1OerDecodeBuffer buffer) [virtual]`

Decode an ASN.1 OBJECT IDENTIFIER that was encoded according to the Octet Encoding Rules (OER).

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.4 `override void Decode (Asn1PerDecodeBuffer buffer) [virtual]`

This method decodes an ASN.1 object identifier value using the packed encoding rules (PER).

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.5 **override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]**

This method decodes an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object
explicitTagging Flag indicating element is explicitly tagged
implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.6 **virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]**

This method decodes an ASN.1 object identifier value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded
attrs Attributes string from element tag

Reimplemented in [Asn1RelativeOID](#).

3.35.3.7 **override void DecodeXML (System.String *buffer*, System.String *attrs*)**

This method decodes an ASN.1 object identifier value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded
attrs Attributes string from element tag

Reimplemented in [Asn1RelativeOID](#).

3.35.3.8 **override void Encode (Asn1PerOutputStream *outs*) [virtual]**

This method encodes an ASN.1 object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

Also throws any exception thrown by the underlying `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.9 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.10 virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode this object identifier to JSON.

3.35.3.11 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an ASN.1 object identifier value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.12 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes an ASN.1 object identifier value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.13 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

This method encodes an ASN.1 OBJECT IDENTIFIER according to Octet Encoding Rules (OER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.14 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an ASN.1 object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.15 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length of encoded component in octets

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.3.16 **override bool Equals (System.Object *value*)**

This method compares this object identifier to the given one for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1ObjectIdentifier](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.35.3.17 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.35.3.18 **override System.String ToString ()**

This method will return a string representation of the OID value. The format is the ASN.1 value format for this type. Note that textual subidentifiers are not used, only numeric values..

Returns

Stringified representation of the value

3.35.3.19 **String ToXMLValue ()**

Return the XML value representation (defined in X.680) for this object identifier.

Exceptions

[Asn1InvalidObjectIDException](#) if this is not a valid value.

3.35.3.20 **virtual void Validate () [protected, virtual]**

Do some minimal validation. Subclasses may override this to adjust for their validation rules (e.g. [Asn1RelativeOID](#) overrides this to be more lax). Minimally, the implementation should enforce that value is not null and that there is at least one arc.

Exceptions

[Asn1InvalidObjectIDException](#) if validation fails

Reimplemented in [Asn1RelativeOID](#).

3.35.4 **Member Data Documentation**

3.35.4.1 **new readonly Asn1Tag _TAG [static]**

The _TAG constant describes the universal tag for this data type (UNIVERSAL 6).

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

3.35.4.2 **const int MAXSUBIDS = 128**

The MAXSUBIDS constant specifies the maximum number of subidentifiers that can appear in an OID value (128).

3.35.4.3 `int [] mValue`

The `mValue` public member variable is where the object identifier value is stored.

3.36 Asn1OctetString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Inherited by [Asn1OpenType](#).

Public Member Functions

- [Asn1OctetString](#) (System.String value)
- [Asn1OctetString](#) (byte[] data, int offset, int nbytes)
- [Asn1OctetString](#) (byte[] data)
- [Asn1OctetString](#) ()
- virtual int [CompareTo](#) (System.Object octstr)
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- void [Decode](#) (Asn1MderDecodeBuffer buffer, int constrainedLength)
- override void [Decode](#) (Asn1MderDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [DecodeContent](#) (Asn1OerDecodeBuffer buffer, int numOctets)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs, bool base64)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- virtual void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix, bool base64)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- void [Encode](#) (Asn1MderOutputStream outs, int constrainedLength)
- override void [Encode](#) (Asn1MderOutputStream outs)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- void [EncodeContent](#) (Asn1OerEncodeBuffer buffer)
- override bool [Equals](#) (System.Object value)
- bool [Equals](#) (String s)
- bool [Equals](#) (byte[] value)
- override int [GetHashCode](#) ()
- int [GetMderLength](#) ()
- virtual System.IO.Stream [toInputStream](#) ()
- override System.String [ToString](#) ()

Static Public Member Functions

- static System.String [EncodeBase64Binary](#) (byte[] data)

Public Attributes

- byte[] [mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Properties

- override int [Length](#) [get]

3.36.1 Detailed Description

This is a container class for holding the components of an ASN.1 octet string value.

3.36.2 Constructor & Destructor Documentation

3.36.2.1 [Asn1OctetString \(\)](#)

This constructor creates an empty octet string that can be used in a Decode method call to receive an octet string value.

3.36.2.2 [Asn1OctetString \(byte\[\] data\)](#)

This constructor initializes an octet string from the given byte array.

Parameters

data Byte array containing an octet string in binary form.

3.36.2.3 [Asn1OctetString \(byte\[\] data, int offset, int nbytes\)](#)

This constructor initializes an octet string from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

Parameters

data Byte array containing an octet string in binary form.

offset The offset in array at which to begin copy.

nbytes Number of bytes to copy from target array

3.36.2.4 [Asn1OctetString \(System.String value\)](#)

This constructor parses the given ASN.1 value text (either a binary or hex data string) and assigns the values to the internal bit string.

Examples of valid value formats are as follows:

Binary string: '11010010111001'B

Hex string: '0fa56920014abc'H

Char string: 'abcdefg'

Parameters

value The ASN.1 value specification text

3.36.3 Member Function Documentation

3.36.3.1 virtual int CompareTo (System.Object *octstr*) [virtual]

This method compares two [Asn1OctetString](#) objects for equality. The OCTET STRING's are equal if a) all octets are equal, and b) the lengths are the same.

This method is required to implement the Comparable interface used for sorting.

Parameters

octstr [Asn1OctetString](#) to compare

Returns

0 if equal, 1 if this string is greater than supplied string, -1 if this string is less than supplied string.

3.36.3.2 virtual void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 octet string from JSON.

Reimplemented in [Asn1OpenType](#).

3.36.3.3 void Decode (Asn1MderDecodeBuffer *buffer*, int *constrainedLength*)

Decode an octet string from the MDER encoding into this object.

Parameters

constrainedLength The constrained length of the type being encoded. Pass -1 if the type is unconstrained. For a constrained length octet string, exactly that many octets will be read.

3.36.3.4 override void Decode (Asn1MderDecodeBuffer *buffer*) [virtual]

Decode an unconstrained octet string from the MDER encoding into this object.

Reimplemented from [Asn1Type](#).

3.36.3.5 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 octet string using the Octet Encoding Rules (OER).

This implementation expects a length determinant in the encoding. This method may be overridden for ASN.1 types that have a fixed length to invoke DecodeContent with the predetermined length.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1OpenType](#).

3.36.3.6 virtual void Decode (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This method decodes a sized ASN.1 octet string value using the packed encoding rules (PER).

Parameters

buffer Decode message buffer object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

3.36.3.7 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 octet string value using the packed encoding rules (PER). The string is assumed to not contain a size constraint.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.8 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 octet string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.9 void DecodeContent (Asn1OerDecodeBuffer *buffer*, int *numOctets*)

This method decodes an ASN.1 OCTET STRING's content that was encoded according to OER, given the total number of octets. This does not decode a length determinant.

Parameters

buffer Decode message buffer object

numOctets Total number of octets encoding the content.

3.36.3.10 virtual void DecodeXER (System.String *buffer*, System.String *attrs*, bool *base64*) [virtual]

This method decodes ASN.1 octet string type using the XML encoding rules (XER). Extended-XER is supported by the base64

Parameters

- buffer* String containing data to be decoded
- attrs* Attributes string from element tag
- base64* pass true if encoding is base64 (extended-XER only)

3.36.3.11 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 octet string type using the XML schema encoding rules(asn2xsd).

Parameters

- buffer* String containing data to be decoded
- attrs* Attributes string from element tag

3.36.3.12 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, long *upper*) [virtual]

This method encodes a size-constrained ASN.1 octet string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

- outs* PER Output Stream object
- lower* Lower bound (inclusive) of size constraint
- upper* Upper bound (inclusive) of size constraint

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.36.3.13 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an unconstrained ASN.1 octet string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

- outs* PER Output Stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.14 **override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]**

This method encodes and writes to the stream an ASN.1 octet string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object
explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.15 **virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]**

Encode this octet string to JSON.

Parameters

outstream

Reimplemented in [Asn1OpenType](#).

3.36.3.16 **virtual void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*, bool *base64*) [virtual]**

This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object
elemName XML element name used to wrap string
nsPrefix Element namespace value
base64 Pass true to encode as base64 (extended-XER only, including XSD compilation)

3.36.3.17 **override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]**

This method encodes ASN.1 octet string type as an xmlhstring.

Parameters

buffer Encode message buffer object
elemName XML element name used to wrap string
nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

3.36.3.18 **override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]**

This method encodes ASN.1 octet string type using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object
elemName XML element name used to wrap string

Reimplemented from [Asn1Type](#).

3.36.3.19 **void Encode (Asn1MderOutputStream *outs*, int *constrainedLength*)**

Encode this octet string into the MDER encoding.

Parameters

constrainedLength The constrained length of the type being encoded. Pass -1 if the type is unconstrained.

Exceptions

[Asn1ConsVioException](#) if a constrained length is given and the value is not exactly that length.
[Asn1MderUnsupported](#) if the length is unconstrained and the actual length > 65535 (maximum allowed by MDER).

3.36.3.20 **override void Encode (Asn1MderOutputStream *outs*) [virtual]**

Encode this octet string into the MDER encoding. This should be used for octet string types that are not of fixed length.

Exceptions

[Asn1MderUnsupported](#) if the actual length > 65535 (maximum allowed by MDER).

Reimplemented from [Asn1Type](#).

3.36.3.21 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

This method encodes this ASN.1 OCTET STRING value, according to OER. This encodes the length determinant, assuming that the OCTET STRING is not fixed-size constrained. Subclasses may override this to simply call EncodeContent for fixed-size constrained strings.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.22 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]**

This method encodes a size-constrained ASN.1 octet string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Parameters

buffer Encode message buffer object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

3.36.3.23 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an unconstrained ASN.1 octet string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.24 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 octet string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length of encoded component

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

3.36.3.25 **override void EncodeAttribute (Asn1XmlEncoder *buffer*, System.String *attrName*) [virtual]**

This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName XML element name used to wrap string

attribute Element attribute value

Reimplemented from [Asn1Type](#).

3.36.3.26 **static System.String EncodeBase64Binary (byte[] *data*) [static]**

Encodes xsd:Base64Binary data into ASCII character using the algorithm defined in RFC2045, as defined in w3c standard <http://www.w3.org/tr/2001/rec-xmlschema-2-20010502#base64Binary>

Parameters

base64binary Array containing base64binary

Returns

Encoded ASCII string

3.36.3.27 void EncodeContent (Asn1OerEncodeBuffer *buffer*)

This method encodes the content of an ASN.1 OCTET STRING, according to OER. (The length determinant is not encoded.)

Parameters

buffer Encode message buffer object

3.36.3.28 override bool Equals (System.Object *value*)

This method compares this octet string value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1OctetString](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.36.3.29 bool Equals (String *s*)

This method compares the given ASN.1 OCTET STRING value to this OCTET STRING value for equality.

Parameters

value The ASN.1 OCTET STRING value to compare with the current Object. Examples of valid value formats are as follows: Binary string: '11010010111001'B Hex string: '0fa56920014abc'H

Returns

`true` if the specified value is equal to this value; otherwise, `false`.

3.36.3.30 bool Equals (byte[] *value*)

This method compares this octet string value to the given value for equality.

Parameters

value The byte array to compare with the current Object.

Returns

`true` if the specified byte array is equal to the current Object; otherwise, `false`.

3.36.3.31 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.36.3.32 **int GetMderLength ()**

Return the length of the MDER encoding for this type, assuming that the type is NOT a fixed length OCTET STRING (if it is fixed length, you need not call this method!).

3.36.3.33 **virtual System.IO.Stream toInputStream () [virtual]**

This method will return a byte array input stream representation of the octet string value.

Returns

Reference to System.IO.MemoryStream object

3.36.3.34 **override System.String ToString ()**

This method will return a string representation of the octet string value. The format is the ASN.1 value format for this type..

Returns

Stringified representation of the value

Reimplemented in [Asn1OpenType](#).

3.36.4 **Member Data Documentation**

3.36.4.1 **new readonly Asn1Tag _TAG [static]**

The _TAG constant describes the universal tag for this data type (UNIVERSAL 4).

Reimplemented from [Asn1Type](#).

3.36.4.2 **byte [] mValue**

This variable holds the octet string value. These are the octets that are encoded when encode is invoked. It is also where the decoded octet string is stored after a Decode operation.

3.36.5 **Property Documentation**

3.36.5.1 **override int Length [get]**

Gets the length of the OCTET STRING in octets.

Value: length of the octet string
Reimplemented from [Asn1Type](#).

3.37 Asn1OpenExt Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeComponent](#) (Asn1BerDecodeBuffer buffer)
- virtual void [DecodeEventComponent](#) (Asn1BerDecodeBuffer buffer)
- void [DecodeExtension](#) (Asn1JsonDecodeBuffer buffer, String name)
- [Asn1OpenType DecodeOpenType](#) (Asn1OerDecodeBuffer buffer, bool present, int index)
- virtual [Asn1OpenType DecodeOpenType](#) (Asn1PerDecodeBuffer buffer, bool present, int index)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- void [Encode](#) (Asn1JsonOutputStream outstream)
- override void [Encode](#) (Asn1XmlEncoder buffer)
- override void [Encode](#) (Asn1XerEncoder buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [EncodeExtBits](#) (Asn1PerEncodeBuffer buffer)
- void [EncodeExtBits](#) (Asn1OerEncodeBuffer buffer)
- bool [HasPresentExtensions](#) ()
- virtual void [SetOpenType](#) ([Asn1OpenType](#) obj, int index)
- virtual void [ShrinkArray](#) (int numrecs)
- override System.String [ToString](#) ()

Public Attributes

- System.Collections.ArrayList [mValue](#)

Properties

- override string [AsnTypeName](#) [get]

3.37.1 Detailed Description

This is a container class for holding open type elements that may occur within an open type extension (i.e. a ... at the end of a constructed type or a ..., ... at some other point in a constructed type).

3.37.2 Member Function Documentation

3.37.2.1 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an open type extension in a SEQUENCE or SET construct using the packed encoding rules (PER). This method will capture each extension item in a separate open type object and store it in the `mValue` public member list variable. If optional items are absent, null placeholders will be inserted in the list.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.37.2.2 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length if implicit element

Reimplemented from [Asn1Type](#).

3.37.2.3 virtual void DecodeComponent (Asn1BerDecodeBuffer *buffer*) [virtual]

This method decodes a single component of a BER open type extension by decoding an open type value and appending it to the list of open type objects.

Parameters

buffer Decode message buffer object

3.37.2.4 virtual void DecodeEventComponent (Asn1BerDecodeBuffer *buffer*) [virtual]

This method decodes a single component of a BER open type extension by decoding an open type value and appending it to the list of open type objects, this function also triggers event handler code, with element name "..."

Parameters

buffer Decode message buffer object

3.37.2.5 void DecodeExtension (Asn1JsonDecodeBuffer *buffer*, String *name*)

Decode a single occurrence of an extension from JSON and add an [Asn1OpenType](#) for it to the list of extensions.

Parameters

name The name of the extension element (previously decoded). This becomes a part of the character data held in the [Asn1OpenType](#) object, so that the character data represents a full JSONNamedValue.

3.37.2.6 `Asn1OpenType DecodeOpenType (Asn1OerDecodeBuffer buffer, bool present, int index)`

This method decodes a single open type extension item in a SEQUENCE or SET construct using the octet encoding rules (OER). It will then add the item to the open extension element list.

Parameters

buffer Decode message buffer object

present Flag indicating whether element is present.

index Index of element in the object array. If $index \geq value.Count$, the item is appended to the list (all such values of index thus have the same behavior).

3.37.2.7 `virtual Asn1OpenType DecodeOpenType (Asn1PerDecodeBuffer buffer, bool present, int index) [virtual]`

This method decodes a single open type extension item in a SEQUENCE or SET construct using the packed encoding rules (PER). It will then add the item to the open extension element list.

Parameters

buffer Decode message buffer object

present Flag indicating whether element is present

index Index of element in the object array

Returns

Decoded open type

3.37.2.8 `override void Encode (Asn1PerOutputStream outs) [virtual]`

This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER).

Also throws any exception thrown by the underlying `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.37.2.9 `override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]`

This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER) and writes it into the stream.

Also throws any exception thrown by the underlying `Asn1BerOutputStream`.

Parameters

outs BER Output Stream object

explicitTagging Flag indicating element is explicitly tagged

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.37.2.10 void Encode (Asn1JsonOutputStream *outstream*)

Encode the open extension elements to JSON

3.37.2.11 override void Encode (Asn1XmlEncoder *buffer*) [virtual]

This method encodes an ASN.1 open type extension value using the XML Encoding as specified in the XML schema standard (asn2xsd).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.37.2.12 override void Encode (Asn1XerEncoder *buffer*) [virtual]

This method encodes an ASN.1 open type extension value using the XML Encoding Rules (XER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.37.2.13 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.37.2.14 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating element is explicitly tagged

Returns

Length of encoded component

Reimplemented from [Asn1Type](#).

3.37.2.15 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

This method encodes the ASN.1 open type extensions using Octet Encoding Rules (OER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.37.2.16 **virtual void EncodeExtBits (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an ASN.1 open type extension value bits using the Packed Encoding Rules (PER).

Parameters

buffer Encode message buffer object

3.37.2.17 **void EncodeExtBits (Asn1OerEncodeBuffer *buffer*)**

This method encodes an ASN.1 open type extension value bits using the Packed Encoding Rules (PER).

Parameters

buffer Encode message buffer object

3.37.2.18 **bool HasPresentExtensions ()**

Return true if this contains present extensions. Null values in the value list represent absent extensions.

3.37.2.19 **virtual void SetOpenType (Asn1OpenType *obj*, int *index*) [virtual]**

This method will add the given open type object to the open extension element list at the given index.

Parameters

obj Open type object

index Index in open type list where element is to be placed

3.37.2.20 **virtual void ShrinkArray (int *numrecs*) [virtual]**

This method adjusts the size of the open type component array downward to the given size value.

Parameters

numrecs Number of entries the array should hold

3.37.2.21 **override System.String ToString ()**

This method will return a string representation of the open extension value. The format is the ASN.1 value format for each open type in the extension.

Returns

Stringified representation of the value

3.37.3 **Member Data Documentation**

3.37.3.1 **System.Collections.ArrayList mValue**

Initial value:

```
new System.Collections.ArrayList ()
```

The value is a list of [Asn1OpenType](#) objects. Each of these objects contains a fully encoded extension item.

3.37.4 **Property Documentation**

3.37.4.1 **override string AsnTypeName [get]**

Gets the ASN.1 type name that is associated with this type. The ASN.1 type name is derived from the input specification and cannot be set by users of the class.

Value: The ASN.1 type name for this type.

Reimplemented from [Asn1Type](#).

3.38 Asn1OpenType Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1OctetString](#).

Inherited by [Asn1ChoiceExt](#).

Classes

- class [SaxHandler](#)

Public Member Functions

- [Asn1OpenType](#) (char[] data, int encoding)
- [Asn1OpenType](#) (string data, int encoding)
- [Asn1OpenType](#) ([Asn1EncodeBuffer](#) buffer)
- [Asn1OpenType](#) (byte[] data, int offset, int nbytes, int encoding)
- [Asn1OpenType](#) (byte[] data, int offset, int nbytes)
- [Asn1OpenType](#) (byte[] data, int encoding)
- [Asn1OpenType](#) (byte[] data)
- [Asn1OpenType](#) ()
- override void [Decode](#) ([Asn1JsonDecodeBuffer](#) buffer)
- override void [Decode](#) ([Asn1OerDecodeBuffer](#) buffer)
- override void [Decode](#) ([Asn1PerDecodeBuffer](#) buffer)
- override void [Decode](#) ([Asn1BerDecodeBuffer](#) buffer, bool explicitTagging, int implicitLength)
- void [DecodeExtension](#) ([Asn1JsonDecodeBuffer](#) buffer, String name)
- override void [Encode](#) ([Asn1PerOutputStream](#) outs)
- override void [Encode](#) ([Asn1BerOutputStream](#) outs, bool explicitTagging)
- override void [Encode](#) ([Asn1JsonOutputStream](#) outstream)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer, String elemName)
- override void [Encode](#) ([Asn1XmlEncoder](#) buffer)
- override void [Encode](#) ([Asn1XmlEncoder](#) buffer, String elemName, String nsPrefix)
- override void [Encode](#) ([Asn1OerEncodeBuffer](#) buffer)
- override void [Encode](#) ([Asn1PerEncodeBuffer](#) buffer)
- override int [Encode](#) ([Asn1BerEncodeBuffer](#) buffer, bool explicitTagging)
- void [EncodeAsExtension](#) ([Asn1JsonOutputStream](#) outstream)
- void [EncodeAsExtension](#) ([Asn1XerEncoder](#) buffer)
- void [EncodeAsExtension](#) ([Asn1XmlEncoder](#) buffer)
- char[] [GetCharData](#) ()
- int [GetDataEncoding](#) ()
- virtual [Asn1XerSaxHandler](#) [GetSaxHandler](#) (bool captureOuterElem)
- virtual [Asn1XerSaxHandler](#) [GetSaxHandler](#) ()
- void [SetBinaryData](#) (byte[] data, int encoding)
- void [SetCharData](#) (char[] data, int encoding)
- void [SetCharData](#) (String data, int encoding)
- override System.String [ToString](#) ()

Protected Attributes

- int `dataEncoding`
- internal `Asn1EncodeBuffer mEncodeBuffer`
- internal int `mLength`

Properties

- override string `AsnTypeName` [get]

3.38.1 Detailed Description

This is a container class for holding an ASN.1 open type value.

Where the data is internally stored and how it is interpreted is controlled by a "dataEncoding" field. You can specify the data encoding when creating the object. It will also be set when decoding. The following explains the possible data encodings:

- UNKNOWN: "mValue" is used to hold byte data. It is interpreted as the encoding of the actual value according to some unknown encoding rules (BER, PER, OER, XER, JSON, or some other).
- BER, PER, OER: "mValue" is used to hold byte data. It is interpreted as the encoding of the actual value according to BER, PER, or OER, respectively.
- XER: "mValue" is used to hold byte data. It is interpreted as the the encoding of the actual value in XML (whether X.693 or Obj-Sys rules), where the XML characters are encoded to bytes using the UTF-8 character encoding.
- JSON: a private field is used to hold character data. The character data is the encoding of the actual value in JSON.

Note especially that the data encoding indicates the encoding rules used to encode the actual value. That result is typically encoded as part of some larger encoding. The data encoding does NOT indicate the rules used for that larger encoding. For example, using an xmlhstring representation, XER and JSON will encode an open type that was previously encoded using any arbitrary encoding rules. When decoding such data, the data encoding will be UNKNOWN, not XER nor JSON.

3.38.2 Constructor & Destructor Documentation

3.38.2.1 `Asn1OpenType ()`

This constructor creates an empty type that can be used in a Decode method call to receive an encoded value. The data encoding is UNKNOWN.

3.38.2.2 `Asn1OpenType (byte[] data)`

This constructor initializes an open type from the given byte array. The array is assumed to contain a previously encoded message component. The data encoding is UNKNOWN.

Parameters

data Byte array containing a previously encoded message component.

3.38.2.3 `Asn1OpenType (byte[] data, int encoding)`

This constructor initializes an open type from the given byte array. The array is assumed to contain a previously encoded message component.

Parameters

data Byte array containing a previously encoded value.

encoding The encoding that describes the meaning of data. Any of the encodings other than JSON.

3.38.2.4 `Asn1OpenType (byte[] data, int offset, int nbytes)`

This constructor initializes the open type from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes. The encoding is UNKNOWN.

Parameters

data Byte array containing a previously encoded value.

offset The offset in array at which to begin copy.

nbytes Number of bytes to copy from target array

3.38.2.5 `Asn1OpenType (byte[] data, int offset, int nbytes, int encoding)`

This constructor initializes the open type from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes. /p>

Parameters

data Byte array containing a previously encoded value.

encoding Starting offset in data from which to copy bytes

nbytes Number of bytes to copy from target array

offset The encoding that describes the meaning of data. Any of the encodings other than JSON.

3.38.2.6 `Asn1OpenType (Asn1EncodeBuffer buffer)`

This constructor initializes an open type using an encoded component. This can be used if a header (for example, a ROSE header) is being prepended to a pre-encoded component. The data encoding is derived from the type of [Asn1EncodeBuffer](#) given, and is possibly UNKNOWN.

Parameters

buffer Reference to encode buffer into which component type was encoded.

3.38.2.7 `Asn1OpenType (string data, int encoding)`

Convenience constructor. Same as [Asn1OpenType](#)(data.ToCharArray(), encoding)

3.38.2.8 Asn1OpenType (char[] *data*, int *encoding*)

Create [Asn1OpenType](#) on the given character data. /p>

Parameters

data The character data array.

encoding The encoding. Either XER or JSON. If JSON, data is taken to be the JSON encoding of the actual value. The array is not copied, so beware using it after passing it here. The "mValue" field will be assigned null. If XER, data is taken to be the XER encoding of the actual value. The "mValue" field will be assigned the UTF-8 character encoding of the given characters.

3.38.3 Member Function Documentation

3.38.3.1 override void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 open type value from JSON. If the JSON value was a JSONNestedValue, the data encoding will be JSON, and the data will be character data. If the JSON value was a JSON string (a quoted xmlhstring), the data encoding will be UNKNOWN and the data will be byte data.

For decoding an unknown extension element, use decodeExtension.

Reimplemented from [Asn1OctetString](#).

3.38.3.2 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 open type value using the Octet Encoding Rules (OER). This object will subsequently contain the encoding of the open type (which should be OER).

Reimplemented from [Asn1OctetString](#).

3.38.3.3 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 open type value using the packed encoding rules (PER). The data encoding will be set to PER.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.4 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 open type value. The data encoding will be set to BER.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.5 void DecodeExtension (Asn1JsonDecodeBuffer *buffer*, String *name*)

Decode an extension from JSON. There should be, possibly after some leading whitespace, a JSON value on the input. After decoding, this object's data encoding will be JSON and the character data will be the a JSONNamedValue, formed from the given (previously decoded) name and the decoded value.

3.38.3.6 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER). The data should be the value pre-encoded in PER.

Also throws any exception thrown by the underlying Asn1PerOutputStream.

Parameters

outs PER Output Stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.7 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 open type value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER). The data should be the value pre-encoded in BER.

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.8 override void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode this octet string to JSON.

Parameters

ostream

Reimplemented from [Asn1OctetString](#).

3.38.3.9 override void Encode (Asn1XerEncoder *buffer*) [virtual]

This method encodes an ASN.1 open type value using the XML Encoding Rules (XER).

If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.38.3.10 override void Encode (Asn1XerEncoder *buffer*, String *elemName*)

This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd).

If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters

buffer Encode message buffer object

elemName Ignored

3.38.3.11 override void Encode (Asn1XmlEncoder *buffer*) [virtual]

This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd).

If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.38.3.12 override void Encode (Asn1XmlEncoder *buffer*, String *elemName*, String *nsPrefix*)

This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd). If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters

buffer Encode message buffer object

elemName Ignored

nsPrefix Ignored

3.38.3.13 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

This method encodes an ASN.1 open type value using the Octet Encoding Rules (OER). The data should be the value pre-encoded in OER.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.14 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER). The data should be the value pre-encoded in PER.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.15 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 open type value. The value is assumed to be an already-encoded BER message component and will be copied to the encoded buffer. An optimization is available in which no copy will be performed if the encoded component is already present in the encode buffer. This is done if the form of constructor specifying only a component length is used.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating element is explicitly tagged

Returns

Length of encoded component

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

3.38.3.16 **void EncodeAsExtension (Asn1JsonOutputStream *outstream*)**

This method encodes an extension element to JSON. If the data encoding is other than JSON, it cannot be encoded and the data is dropped.

3.38.3.17 void EncodeAsExtension (Asn1XerEncoder *buffer*)

This method encodes an extension element to XML. If the data encoding is other than XER, its hexadecimal representation is encoded inside an XML comment.

Parameters

buffer

3.38.3.18 void EncodeAsExtension (Asn1XmlEncoder *buffer*)

This method encodes an extension element to XML. If the data encoding is other than XER, the hexadecimal representation of the data is encoded inside an XML comment.

Parameters

buffer Encode message buffer object

3.38.3.19 char [] GetCharData ()

Returns the character data for the open type. Currently, this is only supported when getDataEncoding() returns JSON.

Returns

The char data. The actual internal array is returned; a copy is not made.

Exceptions

RuntimeException if data encoding indicates there is no char data

3.38.3.20 int GetDataEncoding ()

Return the data encoding of the data.

3.38.3.21 virtual Asn1XerSaxHandler GetSaxHandler (bool *captureOuterElem*) [virtual]

This method returns the Asn1XerOpenType.SaxHandler class instance used for ASN.1 XER encoding.

Parameters

captureOuterElem Pass true if the outer element start and end tags (if present) should be captured. The outer element is the element for which startElement is called when the level is the start level. Note that this parameter is ignored if you have previously invoked one of the getSaxHandler methods on this object.

Returns

Asn1XerOpenType.SaxHandler object

3.38.3.22 virtual Asn1XerSaxHandler GetSaxHandler () [virtual]

This method returns the `Asn1XerOpenType.SaxHandler` class instance used for ASN.1 XER encoding. If this is the first invocation of any of the `GetSaxHandler` methods on this object, the returned handler will capture the outer element start/end tags. Otherwise, this will simply return the same SAX handler as previously returned.

Returns

`Asn1XerOpenType.SaxHandler` object

3.38.3.23 void SetBinaryData (byte[] data, int encoding)

Sets the binary data for the open type and assigns the data encoding to the given encoding.

Parameters

data The binary data that make up an encoding of the actual value.

encoding Identifies the encoding rules for which data is an encoding. This can be any of the encodings except JSON.

3.38.3.24 void SetCharData (char[] data, int encoding)

Sets the character data for the open type and assigns the data encoding to the given encoding.

Parameters

data The character data that make up an encoding of the actual value.

encoding Permissible values are JSON and XER. If XER, the given data will be converted to bytes using UTF-8 and held in the "mValue" field. If JSON, the given data will be held as-is. The array will not be copied. The "mValue" field will be assigned null.

3.38.3.25 void SetCharData (String data, int encoding)

Convenience method; same as

```
SetCharData (mValue.toCharArray(), encoding)
```

3.38.3.26 override System.String ToString ()

This method will return a string representation of the open type value. If the data encoding is XER or JSON, the string will consist of the corresponding characters (in the case of XER, the byte data is converted to characters using UTF-8). In all other cases, the hexadecimal representation of the byte data is returned.

Returns

Stringified representation of the value

Reimplemented from [Asn1OctetString](#).

3.38.4 Member Data Documentation

3.38.4.1 `int dataEncoding` [protected]

Specifies the nature of the data held by this object. /p>

3.38.4.2 `internal Asn1EncodeBuffer mEncodeBuffer` [protected]

The encode buffer into which component type will be encoded. /p>

3.38.4.3 `internal int mLength` [protected]

Length of the pre-encoded component. /p>

3.38.5 Property Documentation

3.38.5.1 `override string AsnTypeName` [get]

Gets the ASN.1 type name that is associated with this type. The ASN.1 type name is derived from the input specification and cannot be set by users of the class.

Value: The ASN.1 type name for this type.

Reimplemented from [Asn1Type](#).

3.39 Asn1OutputStream Class Reference

Public Member Functions

- [Asn1OutputStream](#) (System.IO.Stream *os*)
- override void [Close](#) ()
- override void [Flush](#) ()
- override int [Read](#) (byte[] buffer, int offset, int count)
- override long [Seek](#) (long offset, System.IO.SeekOrigin origin)
- override void [SetLength](#) (long value)
- override void [Write](#) (System.Byte[] b, int off, int len)
- virtual void [Write](#) (byte[] b)
- void [Write2Bytes](#) (int value)
- void [Write4Bytes](#) (int value)
- override void [WriteByte](#) (byte b)
- virtual void [WriteByte](#) (int b)

Protected Attributes

- internal System.IO.Stream *os*

Properties

- override bool [CanRead](#) [get]
- override bool [CanSeek](#) [get]
- override bool [CanWrite](#) [get]
- Asn1Context [Context](#) [get]
- override long [Length](#) [get]
- override long [Position](#) [get, set]

3.39.1 Detailed Description

This abstract class implements the base output stream to encode ASN.1 messages.

3.39.2 Constructor & Destructor Documentation

3.39.2.1 Asn1OutputStream (System.IO.Stream *os*)

This constructor creates an output stream object.

Parameters

- os* The underlying System.IO.Stream object.

3.39.3 Member Function Documentation

3.39.3.1 override void Close ()

Closes this output stream and releases any system resources associated with this stream. The general contract of `close` is that it closes the output stream. A closed stream cannot perform output operations and cannot be reopened.

Exceptions

System.IO.IOException An error occurred while trying to close the stream.

3.39.3.2 override void Flush ()

Flushes this output stream and forces any buffered output bytes to be written out. The general contract of `flush` is that calling it is an indication that, if any bytes previously written have been buffered by the implementation of the output stream, such bytes should immediately be written to their intended destination.

Exceptions

System.IO.IOException An I/O error occurs.

System.ObjectDisposedException The stream is closed.

3.39.3.3 override int Read (byte[] buffer, int offset, int count)

This method always throws `NotSupportedException`. [Asn1OutputStream](#) doesn't support reading.

Parameters

buffer When this method returns, contains the specified byte array with the values between `offset` and (`offset` + `count` - 1) replaced by the bytes read from the current source.

offset The byte offset in array at which to begin reading.

count The maximum number of bytes to read.

Returns

The total number of bytes read into the buffer.

Exceptions

System.NotSupportedException The stream does not support reading.

3.39.3.4 override long Seek (long offset, System.IO.SeekOrigin origin)

Sets the current position of this stream to the given value.

Parameters

offset The point relative to origin from which to begin seeking.

origin Specifies the beginning, the end, or the current position as a reference point for origin, using a value of type `SeekOrigin`.

Returns

The new position in the stream.

Exceptions

System.IO.IOException An I/O error occurs.

System.NotSupportedException The stream does not support seeking, such as if the `FileStream` is constructed from a pipe or console output.

System.ArgumentException Attempted seeking before the beginning of the stream.

System.ObjectDisposedException Methods were called after the stream was closed.

3.39.3.5 override void SetLength (long value)

Sets the length of this stream to the given value.

Parameters

value The new length of the stream.

Exceptions

System.IO.IOException An I/O error has occurred.

System.NotSupportedException The stream does not support both writing and seeking.

System.ArgumentOutOfRangeException Attempted to set the value parameter to less than 0.

3.39.3.6 override void Write (System.Byte[] b, int off, int len)

Writes `len` bytes from the specified byte array starting at offset `off` to this output stream.

Parameters

b The byte array data to be written.

off The offset in array at which to begin write.

len the number of bytes to write.

Exceptions

System.ArgumentNullException `array` is a null reference

System.ArgumentException `offset` and `count` describe an invalid range in array.

System.ArgumentOutOfRangeException `offset` or `count` is negative.

System.IO.IOException An I/O error occurs.

System.ObjectDisposedException The stream is closed.

System.NotSupportedException The current stream instance does not support writing.

3.39.3.7 virtual void Write (byte[] b) [virtual]

Writes `b.length` bytes from the specified byte array to this output stream. The general contract for `write(b)` is that it should have exactly the same effect as the call `Write(b, 0, b.length)`.

Parameters

b the data.

Exceptions

System.ArgumentNullException array is a null reference

System.ArgumentException offset and count describe an invalid range in array.

System.ArgumentOutOfRangeException offset or count is negative.

System.IO.IOException An I/O error occurs.

System.ObjectDisposedException The stream is closed.

System.NotSupportedException The current stream instance does not support writing.

3.39.3.8 void Write2Bytes (int value)

Write the lowest two bytes of value to the output stream. The lowest byte is written last.

Parameters

value

3.39.3.9 void Write4Bytes (int value)

Write the four bytes of value to the output stream. The lowest byte is written last.

Parameters

value

3.39.3.10 override void WriteByte (byte b)

Writes the specified byte to this output stream. The general contract for `Write` is that one byte is written to the output stream. The byte to be written is the eight low-order bits of the argument `b`. The 24 high-order bits of `b` are ignored.

Parameters

b the byte.

Exceptions

System.ObjectDisposedException The stream is closed.

System.NotSupportedException The stream does not support writing.

3.39.3.11 virtual void WriteByte (int b) [virtual]

Writes the specified byte to this output stream. The general contract for `Write` is that one byte is written to the output stream. The byte to be written is the eight low-order bits of the argument `b`. The 24 high-order bits of `b` are ignored.

Parameters

b the byte.

Exceptions

System.ObjectDisposedException The stream is closed.

System.NotSupportedException The stream does not support writing.

3.39.4 Member Data Documentation

3.39.4.1 internal System.IO.Stream os [protected]

C# Stream object

3.39.5 Property Documentation

3.39.5.1 override bool CanRead [get]

Gets a value indicating whether the current stream supports reading.

Value: false, the stream doesn't supports reading.

3.39.5.2 override bool CanSeek [get]

Gets a value indicating whether the current stream supports seeking.

Value: true if the stream supports seeking; otherwise, false.

3.39.5.3 override bool CanWrite [get]

Gets a value indicating whether the current stream supports writing.

Value: true if the stream supports writing; otherwise, false.

3.39.5.4 Asn1Context Context [get]

The context associated with this output stream.

3.39.5.5 override long Length [get]

Gets the length in bytes of the stream.

Value: A long value representing the length of the stream in bytes.

Exceptions

System.NotSupportedException `CanSeek` for this stream is false.

System.IO.IOException An I/O error occurs, such as the file being closed.

3.39.5.6 override long Position [get, set]

Gets or sets the current position of this stream.

Value: The current position of this stream.

Exceptions

System.NotSupportedException The stream does not support seeking.

System.IO.IOException An I/O error occurs.

System.ArgumentOutOfRangeException Attempted to set the position to a negative value.

System.IO.EndOfStreamException Attempted seeking past the end of a stream that does not support this.

3.40 Asn1PrintableString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn18BitCharString](#).

Public Member Functions

- [Asn1PrintableString](#) (System.String data)
- [Asn1PrintableString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.40.1 Detailed Description

This is a container class for holding the components of an ASN.1 printable string value.

3.40.2 Constructor & Destructor Documentation

3.40.2.1 Asn1PrintableString ()

The default constructor creates an empty string object.

3.40.2.2 Asn1PrintableString (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data value string

3.40.3 Member Function Documentation

3.40.3.1 override void Decode (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength) [virtual]

This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.40.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to stream an ASN.1 printable string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws Exception, if any exception thrown by the underlying System.IO.Stream.

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Reimplemented from [Asn1Type](#).

3.40.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.40.4 Member Data Documentation

3.40.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 19).

Reimplemented from [Asn1Type](#).

3.41 Asn1Real Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1Real](#) (double value)
- [Asn1Real](#) ()
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [DecodeDouble](#) (Asn1OerDecodeBuffer buffer)
- void [DecodeSingle](#) (Asn1OerDecodeBuffer buffer)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs, bool decodingElemName, bool modifiedEncodings)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outstream)
- void [Encode](#) (Asn1XmlEncoder buffer, String elemName, String nsPrefix, bool asText)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- void [EncodeDouble](#) (Asn1OerEncodeBuffer buffer)
- void [EncodeSingle](#) (Asn1OerEncodeBuffer buffer)
- virtual void [EncodeValue](#) (Asn1XmlEncoder buffer)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (double value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

Static Public Member Functions

- static System.String [NormalizedRealValueToString](#) (double value)

Public Attributes

- double [mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Protected Member Functions

- void `Decode` (`Asn1DecodeBuffer` buffer, int length)

3.41.1 Detailed Description

This class represents the ASN.1 REAL built-in type.

3.41.2 Constructor & Destructor Documentation

3.41.2.1 `Asn1Real` ()

The default constructor sets the double value to zero.

3.41.2.2 `Asn1Real` (double *value*)

This constructor creates an REAL object from a double value.

Parameters

value double value

3.41.3 Member Function Documentation

3.41.3.1 virtual void `Decode` (`Asn1JsonDecodeBuffer` *buffer*) [`virtual`]

Decode ASN.1 REAL from JSON.

3.41.3.2 override void `Decode` (`Asn1OerDecodeBuffer` *buffer*) [`virtual`]

Decode a REAL value, encoded according to OER, into this object.

This method applies to unconstrained REAL values and REAL values that are constrained but not meeting the requirements for encoding using IEEE 754 single or double precision format.

Parameters

buffer

Exceptions

IOException

Reimplemented from [Asn1Type](#).

3.41.3.3 override void `Decode` (`Asn1PerDecodeBuffer` *buffer*) [`virtual`]

This method decodes ASN.1 REAL value using the Packed Encoding Rules (PER). The length and contents components of the message are decoded. The decoded result is stored in the public member 'mValue' in this object.

Parameters

buffer PER Decode message buffer object

Reimplemented from [Asn1Type](#).

3.41.3.4 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 REAL value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.41.3.5 void Decode (Asn1DecodeBuffer *buffer*, int *length*) [protected]

This method decodes the content octets of an ASN.1 REAL value into this object, where the REAL was encoded as for BER and the length is taken to be as given.

Note that this method is used for OER also, since OER uses the same content octets as BER, at least in certain cases.

Parameters

buffer Decode message buffer object

length Length of contents

3.41.3.6 void DecodeDouble (Asn1OerDecodeBuffer *buffer*)

Decode a REAL value, encoded according to OER in double precision format.

Parameters

buffer

Exceptions

IOException

3.41.3.7 void DecodeSingle (Asn1OerDecodeBuffer *buffer*)

Decode a REAL value, encoded according to OER in single precision format.

Parameters

buffer

Exceptions

IOException

3.41.3.8 virtual void DecodeXER (System.String *buffer*, System.String *attrs*, bool *decodingElemName*, bool *modifiedEncodings*) [virtual]

This method decodes an ASN.1 real value using XER.

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

decodingElemName Pass true if you the ASN.1 value being decoded was encoded as an empty element and *buffer* is the element name. Such an encoding occurs for the special real values under basic-XER, canonical-XER, and extended-XER without GLOBAL-DEFAULTS MODIFIED-ENCODINGS present.

modifiedEncodings Pass TRUE if decoding under extended-XER with GLOBAL-DEFAULTS MODIFIED-ENCODINGS present.

3.41.3.9 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 real value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.41.3.10 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes ASN.1 REAL value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Also throws any exception thrown by the *Asn1PerOutputStream*.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.41.3.11 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 real value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.41.3.12 virtual void Encode (Asn1JsonOutputStream *outstream*) [virtual]

Encode this ASN.1 real value to JSON.

Parameters

outstream

3.41.3.13 void Encode (Asn1XmlEncoder *buffer*, String *elemName*, String *nsPrefix*, bool *asText*)

This method encodes an ASN.1 real value according to XER encoding rules. It is for use with extended-XER.

Parameters

buffer Encode message buffer object

elemName Element name

asText If TRUE, encode special values as text. Otherwise, special values are encoded as empty elements.

3.41.3.14 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an ASN.1 real value according to Obj-Sys encoding rules. The value is encoded as text.

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

3.41.3.15 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes an ASN.1 real value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.41.3.16 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

Encode this REAL value, according to OER, into the buffer. This method applies to unconstrained REAL values and REAL values that are constrained but not meeting the requirements for encoding using IEEE 754 single or double precision format.

Reimplemented from [Asn1Type](#).

3.41.3.17 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes ASN.1 REAL value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

Reimplemented from [Asn1Type](#).

3.41.3.18 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 REAL value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1Type](#).

3.41.3.19 **override void EncodeAttribute (Asn1XmlEncoder *buffer*, System.String *attrName*) [virtual]**

This method encodes an ASN.1 real value using the XML Encoding as specified in the W3C XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

attrName Attribute name

Reimplemented from [Asn1Type](#).

3.41.3.20 **void EncodeDouble (Asn1OerEncodeBuffer *buffer*)**

Encode this REAL value, according to OER, in double precision format, into the buffer.

Parameters

buffer

Exceptions

IOException

3.41.3.21 void EncodeSingle (Asn1OerEncodeBuffer *buffer*)

Encode this REAL value, according to OER, in single precision format, into the buffer.

3.41.3.22 virtual void EncodeValue (Asn1XmlEncoder *buffer*) [virtual]

This method encodes an ASN.1 real value using the XML encoding (non-XER).

Parameters

buffer Encode message buffer object

3.41.3.23 override bool Equals (System.Object *value*)

Determines whether the specified Object is equal to the current Object.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1Real](#).

Returns

true if the specified Object is equal to the current Object; otherwise, false.

3.41.3.24 virtual bool Equals (double *value*) [virtual]

This method compares this REAL value to the given value for equality.

Parameters

value The double value to compare with the current Object.

Returns

true if the specified double value is equal to the current Object; otherwise, false.

3.41.3.25 override int GetHashCode ()

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.41.3.26 `static System.String NormalizedRealValueToString (double value) [static]`

This method will return a string representation of the normalized REAL value.

The output format is value format [-]X.XXXXXE[-]XXX.

The format is the ASN.1 value format for this type. This means it is a "NumericRealValue" as defined in X.680. Additionally, if there is a leading minus sign, there will be no whitespace between it and the first digit of the integer part, making it also an "XMLNumericRealValue".

Parameters

value value to be normalized and stringified.

Returns

the string as described above

3.41.3.27 `override System.String ToString ()`

This method will return a string representation of the REAL value. The format is the ASN.1 value format for this type..

Returns

Stringified representation of the value

3.41.4 Member Data Documentation

3.41.4.1 `new readonly Asn1Tag _TAG [static]`

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 9).

Reimplemented from [Asn1Type](#).

3.41.4.2 `double mValue`

This public member variable is where the double value is stored. This is the value that is encoded when one of the encode methods is called. It is also where the decoded result is stored when a Decode method is called.

3.42 Asn1Real10 Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1UTF8String](#).

Public Member Functions

- [Asn1Real10](#) (System.String data)
- [Asn1Real10](#) ()
- void [ConvertToDecimal](#) ()
- void [ConvertToNR3Form](#) (bool cxeForm)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1CerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- virtual int [Encode](#) (Asn1DerEncodeBuffer buffer, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- byte [GetNumberForm](#) ()

Static Public Member Functions

- static byte [GetNumberForm](#) (System.String value)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.42.1 Detailed Description

This class represents the ASN.1 REAL BASE 10 built-in type.

3.42.2 Constructor & Destructor Documentation

3.42.2.1 [Asn1Real10](#) ()

The default constructor sets the real10 value to zero.

3.42.2.2 [Asn1Real10](#) (System.String data)

This constructor creates an real10 object from a string value.

Parameters

data String value

3.42.3 Member Function Documentation

3.42.3.1 void ConvertToDecimal ()

This method convert the contained real10 value to XML decimal. Result number placed in mStringBuffer field.

3.42.3.2 void ConvertToNR3Form (bool *cxerForm*)

This method convert the contained real10 value to NR3 form. NR3 form number placed in mStringBuffer field.

3.42.3.3 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 REAL value, having base 10, that was encoded according to OER.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1UTF8String](#).

3.42.3.4 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an real10 value using the Packed Encoding Rules (PER). The length and contents components of the message are decoded. The decoded result is stored in the public member 'value' in this object.

Parameters

buffer PER Decode message buffer object

Returns

void. Decoded result is stored in the 'value' member variable and can be accessed using '<object>.value'.

Reimplemented from [Asn1UTF8String](#).

3.42.3.5 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1UTF8String](#).

3.42.3.6 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an ASN.1 real10 value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

outs PER Encode message buffer object

<throws> IOException Any exception thrown by the Asn1PerOutputStream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Reimplemented from [Asn1UTF8String](#).

3.42.3.7 override void Encode (Asn1CerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Canonical Encoding Rules (CER).

Parameters

outs CER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Reimplemented from [Asn1Type](#).

3.42.3.8 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Reimplemented from [Asn1UTF8String](#).

3.42.3.9 override void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix) [virtual]

This method encodes an ASN.1 real10 value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name
nsPrefix Element namespace value

Reimplemented from [Asn1CharString](#).

3.42.3.10 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes an ASN.1 real10 value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object
elemName Element name

Reimplemented from [Asn1CharString](#).

3.42.3.11 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 REAL value, whose base is 10, according to OER.

Parameters

buffer Encode message buffer object
explicit Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1UTF8String](#).

3.42.3.12 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an real10 value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters

buffer PER Encode message buffer object

Returns

Length of component or negative status value

Reimplemented from [Asn1UTF8String](#).

3.42.3.13 virtual int Encode (Asn1DerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Distinguished Encoding Rules (DER).

Parameters

buffer Encode message buffer object
explicitTagging Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

3.42.3.14 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

buffer Encode message buffer object
explicitTagging Flag indicating explicit tagging should be done

Returns

Length of component or negative status value

Reimplemented from [Asn1UTF8String](#).

3.42.3.15 override void EncodeAttribute (Asn1XmlEncoder *buffer*, System.String *attrName*) [virtual]

This method encodes an ASN.1 real10 value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object
elemName Element name
attribute Element attribute value

Reimplemented from [Asn1Type](#).

3.42.3.16 byte GetNumberForm ()

This method calculates the number form of the contained real10 value.

Parameters

value Real10 value in which to count bits.

Returns

Number form.

3.42.3.17 static byte GetNumberForm (System.String *value*) [static]

This method calculates the number form of an real10 value.

Parameters

value Real10 value in which to count bits.

Returns

Number form.

3.42.4 Member Data Documentation

3.42.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 9).

Reimplemented from [Asn1UTF8String](#).

3.43 Asn1RelativeOID Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier](#).

Public Member Functions

- [Asn1RelativeOID](#) (int[] value)
- [Asn1RelativeOID](#) ()
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- new static readonly [Asn1Tag_TAG](#)

Protected Member Functions

- override void [Validate](#) ()

3.43.1 Detailed Description

This is a container class for holding the components of an ASN.1 relative object identifier value.

3.43.2 Constructor & Destructor Documentation

3.43.2.1 [Asn1RelativeOID](#) ()

This constructor creates an empty object identifier that can be used in a [Decode](#) method call to receive an OID value.

3.43.2.2 [Asn1RelativeOID](#) (int[] value)

This constructor initializes the object identifier from the given array of integer subidentifier values.

Parameters

value Array of subidentifiers

3.43.3 Member Function Documentation

3.43.3.1 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode an ASN.1 RELATIVE-OID that was encoded according to the Octet Encoding Rules (OER).

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.2 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 relative object identifier value using the packed encoding rules (PER).

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.3 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 relative object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.4 override void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 RELATIVE-OID value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.5 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 RELATIVE-OID value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.6 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an ASN.1 relative object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.7 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.8 override void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix) [virtual]

This method encodes an ASN.1 RELATIVE-OID value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.9 override void Encode (Asn1XerEncoder buffer, System.String elemName) [virtual]

This method encodes an ASN.1 RELATIVE-OID value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.10 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 RELATIVE-OID according to Octet Encoding Rules (OER).

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.11 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 relative object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.12 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 relative object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length of encoded component in octets

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.3.13 override void Validate () [protected, virtual]

Do some minimal validation. Subclasses may override this to adjust for their validation rules (e.g. [Asn1RelativeOID](#) overrides this to be more lax). Minimally, the implementation should enforce that value is not null and that there is at least one arc.

Exceptions

[Asn1InvalidObjectIDException](#) if validation fails

Reimplemented from [Asn1ObjectIdentifier](#).

3.43.4 Member Data Documentation

3.43.4.1 new static readonly Asn1Tag_TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 13).

Reimplemented from [Asn1ObjectIdentifier](#).

3.44 Asn1SeqOrderException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1SeqOrderException \(\)](#)

3.44.1 Detailed Description

This class defines the 'ASN.1 sequence order' exception that is thrown when an element is received in a SEQUENCE construct that is not in the correct order..

3.44.2 Constructor & Destructor Documentation

3.44.2.1 Asn1SeqOrderException ()

This constructor creates an exception object with a textual message describing the element that was received out-of-order.

3.45 Asn1Status Class Reference

Public Attributes

- const int `INDEFLEN` = - 9999

3.45.1 Detailed Description

This class defines common constants used in the run-time and generated code. Note that all error reporting in the C# version is done via exceptions. Therefore, there are very few status values defined in the class.

3.45.2 Member Data Documentation

3.45.2.1 const int INDEFLEN = - 9999

This constant indicates an indefinite length field was parsed

3.46 Asn1T61String Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString](#).

Public Member Functions

- [Asn1T61String](#) (System.String data)
- [Asn1T61String](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.46.1 Detailed Description

This is a container class for holding the components of an ASN.1 teletex (T61) string value.

3.46.2 Constructor & Destructor Documentation

3.46.2.1 [Asn1T61String](#) ()

The default constructor creates an empty string object.

3.46.2.2 [Asn1T61String](#) (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data String value of T61String

3.46.3 Member Function Documentation

3.46.3.1 override void [Decode](#) (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 T61String value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.46.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 T61String value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.46.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 T61String type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.46.4 Member Data Documentation

3.46.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 20).

Reimplemented from [Asn1Type](#).

3.47 Asn1Tag Class Reference

Public Member Functions

- [Asn1Tag](#) (short tagclass, short form, int idCode)
- [Asn1Tag](#) ()
- bool [Equals](#) ([Asn1Tag](#) tag)
- virtual bool [Equals](#) (short tagclass, short form, int idCode)
- virtual bool [IsEOC](#) ()
- override System.String [ToString](#) ()

Public Attributes

- const short [APPL](#) = (short) (0x40)
- const short [Bit8Mask](#) = (short) (0x80)
- const short [ClassMask](#) = (short) (0xC0)
- const short [CONS](#) = (short) (0x20)
- const short [CTXT](#) = (short) (0x80)
- const bool [EXPL](#) = true
- const short [EXTIDCODE](#) = (short) (0x1F)
- const short [FormMask](#) = (short) (0x20)
- const short [IDMask](#) = (short) (0x1F)
- const bool [IMPL](#) = false
- const short [L7BitsMask](#) = (short) (0x7f)
- short [mClass](#)
- short [mForm](#)
- int [mIDCode](#)
- const short [PRIM](#) = (short) (0x00)
- const short [PRIV](#) = (short) (0xC0)
- const short [UNIV](#) = (short) (0x00)

Static Public Attributes

- static readonly [Asn1Tag](#) [ENUM](#)
- static readonly [Asn1Tag](#) [EOC](#)
- static readonly [Asn1Tag](#) [SEQUENCE](#)
- static readonly [Asn1Tag](#) [SET](#)

Properties

- virtual bool [Constructed](#) [get]

3.47.1 Detailed Description

This is a container class for holding the components of an ASN.1 tag value.

3.47.2 Constructor & Destructor Documentation

3.47.2.1 Asn1Tag ()

The default constructor initializes all fields to zero

3.47.2.2 Asn1Tag (short tagclass, short form, int idCode)

This constructor initializes all fields to the given values

Parameters

tagclass Tag class value (UNIV, APPL, CTXT, or PRIV)

form Tag form value (PRIM or CONS)

idCode Tag identifier code

3.47.3 Member Function Documentation

3.47.3.1 bool Equals (Asn1Tag tag)

This method compares this tag with the given tag value for equality.

Parameters

tag [Asn1Tag](#) object to which this tag is to be compared

Returns

`true` if the specified Tags are equal; otherwise, `false`.

3.47.3.2 virtual bool Equals (short tagclass, short form, int idCode) [virtual]

This method compares this tag with the given tag value for equality.

Parameters

tagclass Tag class value (UNIV, APPL, CTXT, or PRIV)

form Tag form value (PRIM or CONS)

idCode Tag identifier code

Returns

`true` if the specified Tags are equal; otherwise, `false`.

3.47.3.3 virtual bool IsEOC () [virtual]

This method tests if the tag is an end-of-contents (EOC) tag.

Returns

True if tag is an EOC.

3.47.3.4 **override System.String ToString ()**

This method will return a formatted string representing the tag value. The form is "[<class> <ID>]" (i.e. the ASN.1 standard syntax for a tag value).

Returns

Formatted tag string

3.47.4 **Member Data Documentation**

3.47.4.1 **const short APPL = (short) (0x40)**

Mask value for an APPLICATION tag

3.47.4.2 **const short Bit8Mask = (short) (0x80)**

Bit 8 (MSB) octet mask value

3.47.4.3 **const short ClassMask = (short) (0xC0)**

Mask value to mask the class bits from a tag

3.47.4.4 **const short CONS = (short) (0x20)**

Mask value for CONSTRUCTED form

3.47.4.5 **const short CTXT = (short) (0x80)**

Mask value for a context-specific tag

3.47.4.6 **readonly Asn1Tag ENUM [static]**

ASN.1 ENUMERATED type tag value

3.47.4.7 **readonly Asn1Tag EOC [static]**

ASN.1 end-of-contents (EOC) type tag value

3.47.4.8 **const bool EXPL = true**

This specifies that explicit tagging should be used.

3.47.4.9 **const short EXTIDCODE = (short) (0x1F)**

Mask value for extended tag identifier indicator

3.47.4.10 const short FormMask = (short) (0x20)

Mask value to mask the form bit from a tag

3.47.4.11 const short IDMask = (short) (0x1F)

Mask value to mask the tag ID bits from a tag

3.47.4.12 const bool IMPL = false

This specifies that implicit tagging should be used.

3.47.4.13 const short L7BitsMask = (short) (0x7f)

Lower 7 bits octet mask value

3.47.4.14 short mClass

Tag class value (UNIV, APPL, CTXT, or PRIV)

3.47.4.15 short mForm

Tag form value (PRIM or CONS)

3.47.4.16 int mIDCode

Tag ID code

3.47.4.17 const short PRIM = (short) (0x00)

Mask value for PRIMITIVE form

3.47.4.18 const short PRIV = (short) (0xC0)

Mask value for a PRIVATE tag

3.47.4.19 readonly Asn1Tag SEQUENCE [static]

ASN.1 SEQUENCE type tag value

3.47.4.20 readonly Asn1Tag SET [static]

ASN.1 SET type tag value

3.47.4.21 const short UNIV = (short) (0x00)

Mask value for a UNIVERSAL tag

3.47.5 Property Documentation

3.47.5.1 virtual bool Constructed [get]

Gets the tag is constructed.

Value: `true` if tag is constructed; otherwise `false`.

3.48 Asn1Time Class Reference

This class is used for all TIME types that are not one of the useful time types (viz.

Inherits [Com::Objsys::Asn1::Runtime::Asn18BitCharString](#).

Public Member Functions

- [Asn1Time](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outstr, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Public Attributes

- const int [January](#) = 1

3.48.1 Detailed Description

DATE, DATE-TIME, TIME-OF-DAY, DURATION).

3.48.2 Constructor & Destructor Documentation

3.48.2.1 Asn1Time ()

The default constructor creates an empty string object.

3.48.3 Member Function Documentation

3.48.3.1 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 TIME string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.48.3.2 override void Encode (Asn1BerOutputStream *outstr*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 TIME.

value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters

oustr BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Reimplemented from [Asn1Type](#).

3.48.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 TIME string type.

The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.48.4 Member Data Documentation

3.48.4.1 const int January = 1

Month constants.

These were defined in what used to be [Asn1Time](#) and was renamed to `Asn1AbstractTime`. They are here to allow backward compatibility so users don't have to modify their code to use `Asn1AbstractTime` to refer to these constants.

3.49 Asn1TraceHandler Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1NamedEventHandler](#).

Public Member Functions

- [Asn1TraceHandler](#) (System.IO.StreamWriter ps)
- [Asn1TraceHandler](#) ()
- virtual void [Characters](#) (System.String svalue, short typeCode)
- virtual void [EndElement](#) (System.String name, int index)
- virtual void [StartElement](#) (System.String name, int index)

3.49.1 Detailed Description

This class is a standard named event handler for printing the data in an encoded message in a human-readable format. Note that this handler will work with data encoded using any of the encoding rules (BER, DER, or PER).

3.49.2 Constructor & Destructor Documentation

3.49.2.1 Asn1TraceHandler ()

This constructor sets the output stream to standard output.

3.49.2.2 Asn1TraceHandler (System.IO.StreamWriter ps)

This constructor sets the output stream to the given StreamWriter.

Parameters

ps StreamWriter object

3.49.3 Member Function Documentation

3.49.3.1 virtual void Characters (System.String svalue, short typeCode) [virtual]

The characters callback method is invoked when content (primitive data) is encountered. A stringified representation of the parsed value is returned.

Parameters

svalue Stringified representation of the parsed value. The representation will be in ASN.1 value format.

typeCode Identifier specifying the type of the parsed data variable. The enumerated list of values that might appear here is provided in the the [Asn1Type](#) class (see the documentation on this class for a full list of the names).

Implements [Asn1NamedEventHandler](#).

3.49.3.2 virtual void EndElement (System.String name, int index) [virtual]

The endElement callback method is invoked when the end of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is detected.

Parameters

name Name of the parsed element.

index Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implements [Asn1NamedEventHandler](#).

3.49.3.3 virtual void StartElement (System.String name, int index) [virtual]

The StartElement callback method is invoked when the start of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is encountered.

Parameters

name Name of the parsed element.

index Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implements [Asn1NamedEventHandler](#).

3.50 Asn1Type Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1TypeIF](#).

Inherited by [Asn1BigInteger](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Choice](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1Real](#), and [Asn1UniversalString](#).

Public Member Functions

- void [_SetKey](#) (byte[] rtkey)
- virtual void [Decode](#) (Asn1MderDecodeBuffer buffer)
- virtual void [Decode](#) (System.Object reader, System.IO.Stream byteStream)
- virtual void [Decode](#) (System.Object reader, System.String xmlURI)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1BerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXML](#) (String buffer, String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs)
- virtual void [Encode](#) (Asn1CerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1MderOutputStream buffer, bool useCachedLength)
- virtual void [Encode](#) (Asn1MderOutputStream buffer)
- virtual void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XmlEncoder buffer)
- virtual void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1XerEncoder buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual int [Encode](#) (Asn1BerEncodeBuffer buffer)
- virtual int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- void [EncodeAsOpenType](#) (Asn1OerEncodeBuffer buffer)
- virtual void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- virtual bool [Equals](#) (Asn1Type obj)
- String [GetNonParameterizedTypeName](#) ()
- virtual void [Indent](#) (System.IO.TextWriter outs, int level)
- virtual bool [IsOpenType](#) ()
- virtual bool [MatchTypeName](#) (System.String typeName)
- virtual void [Pdiag](#) (System.String s)
- virtual void [Print](#) (System.String varName)
- virtual void [Print](#) (System.IO.TextWriter outs, System.String varName, int level)
- void [SetNonParameterizedTypeName](#) (String value)
- virtual void [SetOpenType](#) ()

Static Public Member Functions

- static void [_SetKey2](#) (byte[] rtkey)
- static [Asn1Type Decode](#) (Asn1PerDecodeBuffer buffer, Asn1OpenTypeField openTypeField)
- static [Asn1Type Decode](#) (Asn1OerDecodeBuffer buffer, Asn1OpenTypeField openTypeField)
- static [Asn1Type Decode](#) (Asn1BerDecodeBuffer buffer, Asn1OpenTypeField openTypeField, bool explicitTag, int implicitLength)
- static System.String [GetTypeName](#) (short typeCode)

Public Attributes

- const short [BIT_STRING](#) = 3
- const short [BMPString](#) = 30
- const short [BOOLEAN](#) = 1
- const short [DATE](#) = 31
- const short [ENUMERATED](#) = 10
- const short [EOC](#) = 0
- const short [EXTERNAL](#) = 8
- const short [GeneralString](#) = 27
- const short [GeneralTime](#) = 24
- const short [GraphicString](#) = 25
- const short [IA5String](#) = 22
- const short [INTEGER](#) = 2
- const short [NULL](#) = 5
- const short [NumericString](#) = 18
- const short [OBJECT_IDENTIFIER](#) = 6
- const short [ObjectDescriptor](#) = 7
- const short [OCTET_STRING](#) = 4
- const short [OpenType](#) = 99
- const short [PrintableString](#) = 19
- const short [REAL](#) = 9
- const short [RELATIVE_OID_IRI](#) = 36
- const short [RelativeOID](#) = 13
- const short [SEQUENCE](#) = 16
- const short [SET](#) = 17
- const short [T61String](#) = [TeletexString](#)
- const short [TeletexString](#) = 20
- const short [TIME](#) = 14
- const short [UniversalString](#) = 28
- const short [UTCTime](#) = 23
- const short [UTF8String](#) = 12
- const short [VideotexString](#) = 21
- const short [VisibleString](#) = 26

Static Public Attributes

- static readonly [Asn1Tag_TAG](#)

Static Protected Member Functions

- static internal int [MatchTag](#) ([Asn1BerDecodeBuffer](#) buffer, [Asn1Tag](#) tag)
- static internal int [MatchTag](#) ([Asn1BerDecodeBuffer](#) buffer, short tagClass, short tagForm, int tagIDCode)

Properties

- virtual String [AsnTypeName](#) [get]
- virtual int [Length](#) [get]

3.50.1 Detailed Description

This is the base class for all ASN.1 built-in types.

3.50.2 Member Function Documentation

3.50.2.1 void _SetKey (byte[] *rtkey*)

This method is used with the limited run-time to set a run-time key value generated by the compiler to allow the run-time to operate on the licensed hosts. This is not used in the unlimited redistribution versions.

Parameters

rtkey Run-time key generated by ASN1C

3.50.2.2 static void _SetKey2 (byte[] *rtkey*) [static]

This method is used with the limited run-time to set a run-time key value generated by the compiler to allow the run-time to operate on the licensed hosts. This is not used in the unlimited redistribution versions. This is the static version of _SetKey.

Parameters

rtkey Run-time key generated by ASN1C

3.50.2.3 virtual void Decode (Asn1MderDecodeBuffer *buffer*) [virtual]

This method decodes this object's data from an MDER encoding. The implementation for this class throws an exception. When generating MDER code, subclasses will override this implementation.

Parameters

buffer MDER Decode message buffer object

<throws> IOException Any exception thrown by the underlying stream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1OctetString](#).

3.50.2.4 virtual void Decode (System.Object *reader*, System.IO.Stream *byteStream*) [virtual]

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Also throws any IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

Parameters

reader XML reader object

byteStream Input byte stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

3.50.2.5 virtual void Decode (System.Object *reader*, System.String *xmlURI*) [virtual]

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Also throws any IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

Parameters

reader XML reader object

xmlURI URI of a source

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

3.50.2.6 virtual void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) Decode method. It throws an exception because it should never be invoked. Inherited class implements this method in Compiler generated code.

Parameters

buffer PER Encode message buffer object

Exceptions

Asn1Exception if invoked directly

Implements [Asn1TypeIF](#).

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.50.2.7 static Asn1Type Decode (Asn1PerDecodeBuffer *buffer*, Asn1OpenTypeField *openTypeField*) [static]

Decode an open type field value from the given buffer.

Parameters

buffer Decode message buffer object

openTypeField Describes the actual type.

3.50.2.8 virtual void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 value into this object, following the Octet Encoding Rules (OER).

This method MUST be overridden by all subclasses, except for [Asn1Enumerated](#) and subclasses thereof, in order to implement the correct decode behavior for the corresponding ASN.1 type.

This method MUST NOT be invoked on an [Asn1Enumerated](#) object. Such objects are immutable and cannot be decoded into.

This method is used polymorphically when table constraint code is generated and [Asn1Type](#) is used as the declared type for an open type.

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.50.2.9 static Asn1Type Decode (Asn1OerDecodeBuffer *buffer*, Asn1OpenTypeField *openTypeField*) [static]

Decode an open type field value from the given buffer.

Parameters

openTypeField Describes the open type being decoded.

3.50.2.10 static Asn1Type Decode (Asn1BerDecodeBuffer *buffer*, Asn1OpenTypeField *openTypeField*, bool *explicitTag*, int *implicitLength*) [static]

Decode an open type field value from the given buffer.

Parameters

buffer Decode message buffer object

openTypeField Provides the actual type.

explicitTag if true, the value to be decoded is preceded by a tag and length, which must first be decoded. Otherwise, *implicitLength* provides the length of the value to be decoded.

implicitLength The length of the value to be decoded if *explicitTag* was given as false.

3.50.2.11 virtual void Decode (Asn1BerDecodeBuffer *buffer*) [virtual]

This method is used to Decode a message that is encoded in BER or DER format. This version of the method sets tagging to explicit ([Asn1Tag.EXPL](#)) and implicit length to zero.

Parameters

buffer Decode message buffer object

3.50.2.12 virtual void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method is used to Decode a message that is encoded in BER or DER format.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating explicit tag should be parsed from the encoded type.

implicitLength Length of the contents field (only required if explicitTagging is false).

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Time](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.50.2.13 virtual void DecodeXML (String *buffer*, String *attrs*) [virtual]

This method decodes the XML content of a simple type.

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.50.2.14 virtual void Encode (Asn1PerOutputStream *outs*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) encode method using output stream. It throws an exception because it should never be invoked by compiler generated code.

Also throws any exception thrown by the Asn1PerOutputStream.

Parameters

outs PER Output Stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.50.2.15 virtual void Encode (Asn1CerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method writes to the stream an encoded ASN.1 type value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Cer Encoding Rules (CER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs CER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented in [Asn1Real10](#).

3.50.2.16 virtual void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method writes to the stream an encoded ASN.1 type value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Time](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.50.2.17 virtual void Encode (Asn1MderOutputStream buffer, bool useCachedLength) [virtual]

This method encodes this object's data to an MDER encoding. The implementation for this class throws an exception. When generating MDER code, subclasses will override this implementation or else will override the [Encode\(Asn1MderOutputStream\)](#) overload. Do not invoke this function unless you are certain the subclass has overridden it.

Parameters

buffer MDER Encode message buffer object

useCachedLength Indicates whether cached length of encoding should be used. In general, only generated code should pass true.

<throws> IOException Any exception thrown by the underlying stream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

3.50.2.18 virtual void Encode (Asn1MderOutputStream buffer) [virtual]

This method encodes this object's data to an MDER encoding. The implementation for this class invokes [Encode\(buffer, false\)](#). When generating MDER code, subclasses will override this implementation or else will override the [Encode\(Asn1MderOutputStream, bool\)](#) overload. Invoke this method if you are not certain which overload has been overridden by the subclass.

Parameters

buffer MDER Encode message buffer object

<throws> IOException Any exception thrown by the underlying stream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1OctetString](#).

3.50.2.19 virtual void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd). It throws an exception because it should never be invoked. Inherited class implements this method in Compiler generated code.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XML Encode message buffer object

elemName XML element name of item

nsPrefix Element namespace value

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

3.50.2.20 virtual void Encode (Asn1XmlEncoder *buffer*) [virtual]

This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd). This method invokes the generated method with element name and attribute name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XML Encode message buffer object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1OpenExt](#), and [Asn1OpenType](#).

3.50.2.21 virtual void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method is the base implementation of the standard XML Encoding Rules (XER) encode method. It throws an exception because it should never be invoked by compiler generated code.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XER Encode message buffer object

elemName XML element name of item

Exceptions

[Asn1Exception](#) if invoked directly

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

3.50.2.22 virtual void Encode (Asn1XerEncoder *buffer*) [virtual]

This method is the base implementation of the standard XML Encoding Rules (XER) encode method. This method invokes the generated method with element name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XER Encode message buffer object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1OpenExt](#), and [Asn1OpenType](#).

3.50.2.23 virtual void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) encode method. It throws an exception because it should never be invoked. Inherited class implements this method in Compiler generated code.

Parameters

buffer PER Encode message buffer object

Exceptions

[Asn1Exception](#) if invoked directly

Implements [Asn1TypeIF](#).

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.50.2.24 virtual void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

This method encodes the ASN.1 value represented by this object, following the Octet Encoding Rules (OER).

This method MUST be overridden by all subclasses in order to implement the correct encode behavior for the corresponding type.

This method is used polymorphically when table constraint code is generated and [Asn1Type](#) is used as the declared type for an open type.

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.50.2.25 virtual int Encode (Asn1BerEncodeBuffer *buffer*) [virtual]

This method is used to encode a message in BER or DER format. This version of the method sets tagging to explicit ([Asn1Tag.EXPL](#)).

Parameters

buffer Decode message buffer object

Returns

Length of component or negative status value

3.50.2.26 virtual int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method is used to encode this data type in BER or DER format.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tag should be added to the encoded type.

Returns

Length of component or negative status value

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Time](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.50.2.27 void EncodeAsOpenType (Asn1OerEncodeBuffer *buffer*)

This method encodes the ASN.1 value represented by this object, following the Octet Encoding Rules (OER), as the value for the actual type of an open type (i.e., it precedes its encoding with a length).

3.50.2.28 virtual void EncodeAttribute (Asn1XmlEncoder *buffer*, System.String *attrName*) [virtual]

This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd). It throws an exception because it should never be invoked by compiler generated code.

Parameters

buffer XML Encode message buffer object

attrName XML attribute name of item

attribute Element attribute value

<throws> IOException Any exception thrown by the underlying stream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Reimplemented in [Asn1Boolean](#), [Asn1Integer](#), [Asn1OctetString](#), [Asn1Real](#), and [Asn1Real10](#).

3.50.2.29 virtual bool Equals (Asn1Type *obj*) [virtual]

Return true if the two objects are equal. /p> Note: in generated code, we will implement [Equals\(Asn1Type\)](#) rather than Equals(Object) to avoid the need to also implement GetHashCode. The default implementation here is to invoke Equals(Object). This is acceptable for the runtime classes, which have overridden Equals(Object).

3.50.2.30 String GetNonParameterizedTypeName ()

Return this type's NonParameterizedTypeName, if set.

3.50.2.31 static System.String GetTypeName (short *typeCode*) [static]

This method will convert a type code into a type name as defined in the X.680 standard..

Parameters

typeCode Type code to be converted

Returns

ASN.1 type name

3.50.2.32 virtual void Indent (System.IO.TextWriter *outs*, int *level*) [virtual]

This method will indent three spaces in the given print stream. It is used by the print methods to provide a formatted output of an encoded element value.

Parameters

outs Print stream

level Indentation level (no of spaces is 3 x this number)

3.50.2.33 `virtual bool IsOpenType () [virtual]`

Returns open type mode for XML encoding/decoding.

Returns

`true` if open type mode is on.

Implements [Asn1TypeIF](#).

3.50.2.34 `static internal int MatchTag (Asn1BerDecodeBuffer buffer, Asn1Tag tag) [static, protected]`

This method will compare the next parsed tag with the given tag value. If they do not match, an exception will be thrown.

Parameters

buffer Decode message buffer object

tag Tag value to compare

Returns

Decoded length value

Exceptions

Asn1TagMatchFailedException Tag is not equal to expected value

3.50.2.35 `static internal int MatchTag (Asn1BerDecodeBuffer buffer, short tagClass, short tagForm, int tagIDCode) [static, protected]`

This method will compare the next parsed tag with the given tag value. If they do not match, an exception will be thrown.

Parameters

buffer Decode message buffer object

tagClass Tag class value (UNIV, APPL, CTXT, or PRIV)

tagForm Tag form value (PRIM or CONS)

tagIDCode Tag identifier code

Returns

Decoded length value

Exceptions

Asn1TagMatchFailedException Tag is not equal to expected value

3.50.2.36 virtual bool MatchTypeName (System.String typeName) [virtual]

This method is used to check the outer level tag in an XER message to verify it matches the expected value. This method is overridden by generated code. The default implementation always returns true.

Parameters

typeName Type name to compare.

Returns

True if name matches internal name.

3.50.2.37 virtual void Pdiag (System.String s) [virtual]

This is a diagnostics print method. It is a shorthand way to invoke the [Diag](#) object's println method.

Parameters

s diagnostics message to be printed.

3.50.2.38 virtual void Print (System.String varName) [virtual]

This method will format and output a primitive value to the standard console output.

Parameters

varName Name of variable

3.50.2.39 virtual void Print (System.IO.TextWriter outs, System.String varName, int level) [virtual]

This method will format and output a primitive value to the given print stream.

Parameters

outs Print output stream

varName Name of variable

level Indentation level

Implements [Asn1TypeIF](#).

3.50.2.40 void SetNonParameterizedTypeName (String value)

Set this type's NonParameterizedTypeName. The value is specified, based on the ASN.1 type, by X.680.

3.50.2.41 virtual void SetOpenType () [virtual]

Sets open type mode for XML encoding/decoding.

Implements [Asn1TypeIF](#).

3.50.3 Member Data Documentation

3.50.3.1 readonly Asn1Tag_TAG [static]

Initial value:

```
new Asn1Tag(Asn1Tag.UNIV,  
            Asn1Tag.PRIM, Asn1Type.EOC)
```

Will hold tag for possible definitions

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.50.3.2 const short BIT_STRING = 3

BIT_STRING type code = 3

3.50.3.3 const short BMPString = 30

BMPString type code = 30

3.50.3.4 const short BOOLEAN = 1

BOOLEAN type code = 1

3.50.3.5 const short DATE = 31

DATE type code = 31

3.50.3.6 const short ENUMERATED = 10

ENUMERATED type code = 10

3.50.3.7 const short EOC = 0

EOC type code = 0

3.50.3.8 const short EXTERNAL = 8

EXTERNAL type code = 8

3.50.3.9 const short GeneralString = 27

GeneralString type code = 27

3.50.3.10 const short GeneralTime = 24

GeneralTime type code = 24

3.50.3.11 const short GraphicString = 25

GraphicString type code = 25

3.50.3.12 const short IA5String = 22

IA5String type code = 22

3.50.3.13 const short INTEGER = 2

INTEGER type code = 2

3.50.3.14 const short NULL = 5

NULL type code = 5

3.50.3.15 const short NumericString = 18

NumericString type code = 18

3.50.3.16 const short OBJECT_IDENTIFIER = 6

OBJECT_IDENTIFIER type code = 6

3.50.3.17 const short ObjectDescriptor = 7

ObjectDescriptor type code = 7

3.50.3.18 const short OCTET_STRING = 4

OCTET_STRING type code = 4

3.50.3.19 const short OpenType = 99

OpenType type code = 99

3.50.3.20 const short PrintableString = 19

PrintableString type code = 19

3.50.3.21 const short REAL = 9

REAL type code = 9

3.50.3.22 const short RELATIVE_OID_IRI = 36

RELATIVE-OID-IRI type code = 35

3.50.3.23 const short RelativeOID = 13

RELATIVE_OID type code = 13

3.50.3.24 const short SEQUENCE = 16

SEQUENCE type code = 16

3.50.3.25 const short SET = 17

SET type code = 17

3.50.3.26 const short T61String = TeletexString

T61String type code = TeletexString

3.50.3.27 const short TeletexString = 20

TeletexString type code = 20

3.50.3.28 const short TIME = 14

TIME type code = 14

3.50.3.29 const short UniversalString = 28

UniversalString type code = 28

3.50.3.30 const short UTCTime = 23

UTCTime type code = 23

3.50.3.31 const short UTF8String = 12

UTF8String type code = 12

3.50.3.32 const short VideotexString = 21

VideotexString type code = 21

3.50.3.33 const short VisibleString = 26

VisibleString type code = 26

3.50.4 Property Documentation

3.50.4.1 virtual String AsnTypeName [get]

Gets the ASN.1 type name that is associated with this type. The ASN.1 type name is derived from the input specification and cannot be set by users of the class.

Value: The ASN.1 type name for this type.

Reimplemented in [Asn1OpenExt](#), and [Asn1OpenType](#).

3.50.4.2 virtual int Length [get]

Gets the length of types that can be bound by a size constraint (BIT STRING, OCTET STRING, character string, and SEQUENCE OF/SET OF). An attempt to invoke it on any other type will cause an exception to be thrown.

Value: Length of item in units (for example, bits for BIT STRING, octets for OCTET STRING, etc.)

Exceptions

[Asn1InvalidLengthException](#) if called by type rather than size constrained (BIT STRING, OCTET STRING, character string, or SEQUENCE OF/SET OF)

Reimplemented in [Asn1BitString](#), [Asn1CharString](#), [Asn1OctetString](#), and [Asn1UniversalString](#).

3.51 Asn1TypeIF Interface Reference

Inherited by [Asn1Type](#).

Public Member Functions

- void [Decode](#) (Asn1MderDecodeBuffer buffer)
- void [Decode](#) (System.Object reader, System.IO.Stream byteStream)
- void [Decode](#) (System.Object reader, System.String xmlURI)
- void [Decode](#) (Asn1PerDecodeBuffer buffer)
- void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [Encode](#) (Asn1PerOutputStream outs)
- void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- void [Encode](#) (Asn1MderOutputStream buffer)
- void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- void [Encode](#) (Asn1XmlEncoder buffer)
- void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- void [Encode](#) (Asn1XerEncoder buffer)
- void [Encode](#) (Asn1PerEncodeBuffer buffer)
- int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- bool [IsOpenType](#) ()
- void [Print](#) (System.IO.TextWriter outs, System.String varName, int level)
- void [SetOpenType](#) ()

3.51.1 Detailed Description

This is the base interface for all ASN.1 built-in types.

3.51.2 Member Function Documentation

3.51.2.1 void Decode (Asn1MderDecodeBuffer *buffer*)

This method decodes this object's data from an MDER encoding. When MDER code has not been generated, classes implementing this interface may simply throw an exception.

Implemented in [Asn1OctetString](#), and [Asn1Type](#).

3.51.2.2 void Decode (System.Object *reader*, System.IO.Stream *byteStream*)

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Throws an IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

Parameters

reader XML reader object

byteStream Input byte stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Implemented in [Asn1Type](#).

3.51.2.3 void Decode (System.Object *reader*, System.String *xmlURI*)

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Throws an IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

Parameters

reader XML reader object

xmlURI URI of a source

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Implemented in [Asn1Type](#).

3.51.2.4 void Decode (Asn1PerDecodeBuffer *buffer*)

This method declaration is the signature of the standard Packed Encoding Rules (PER) Decode method.

Parameters

buffer PER Encode message buffer object

Implemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.51.2.5 void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*)

This method declaration is the signature of the standard Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) Decode method.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating explicit tag should be parsed from the encoded type.

implicitLength Length of the contents field (only required if explicit is false).

Implemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Time](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.51.2.6 void Encode (Asn1PerOutputStream *outs*)

This method declaration is the signature of the streaming oriented PER encode method.

Also throws any exception thrown by the [Asn1PerOutputStream](#).

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.51.2.7 void Encode (Asn1BerOutputStream outs, bool explicitTagging)

This method declaration is the signature of the streaming oriented BER encode method.

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Time](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.51.2.8 void Encode (Asn1MderOutputStream buffer)

This method encodes this object's data to an MDER encoding. When MDER code has not been generated, classes implementing this interface may simply throw an exception.

Implemented in [Asn1OctetString](#), and [Asn1Type](#).

3.51.2.9 void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)

This method declaration is the signature of the standard XML Encoding as specified in the XML schema standard(asn2xsd).

Also throws any exception thrown by the underlying stream.

Parameters

buffer XML Encode message buffer object

elemName XML element name of item

nsPrefix Element namespace value

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

3.51.2.10 void Encode (Asn1XmlEncoder *buffer*)

This method declaration is the signature of the standard XML Encoding as specified in the XML schema standard(asn2xsd). This method invokes the generated method with element name and attribute name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XML Encode message buffer object

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1OpenExt](#), [Asn1OpenType](#), and [Asn1Type](#).

3.51.2.11 void Encode (Asn1XerEncoder *buffer*, System.String *elemName*)

This method declaration is the signature of the standard XML Encoding Rules (XER) encode method.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XER Encode message buffer object

elemName XML element name of item

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

3.51.2.12 void Encode (Asn1XerEncoder *buffer*)

This method declaration is the signature of the standard XML Encoding Rules (XER) encode method. This method invokes the generated method with element name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

Parameters

buffer XER Encode message buffer object

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1OpenExt](#), [Asn1OpenType](#), and [Asn1Type](#).

3.51.2.13 void Encode (Asn1PerEncodeBuffer *buffer*)

This method declaration is the signature of the standard Packed Encoding Rules (PER) encode method.

Parameters

buffer PER Encode message buffer object

Implemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

3.51.2.14 int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*)

This method declaration is the signature of the standard Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) encode method.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tag should be added to the encoded type.

Returns

Length of component or negative status value

Implemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1Real10](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Time](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

3.51.2.15 bool IsOpenType ()

Returns open type mode for XML encoding/decoding.

Returns

`true` if open type mode is on.

Implemented in [Asn1Type](#).

3.51.2.16 void Print (System.IO.TextWriter *outs*, System.String *varName*, int *level*)

This method declaration is the signature of the standard print method used to print the contents of the object representing the ASN.1 type.

Parameters

outs Output print stream

varName Name of the variable being printed

level Indentation level

Implemented in [Asn1Type](#).

3.51.2.17 void SetOpenType ()

Sets open type mode for XML encoding/decoding.

Implemented in [Asn1Type](#).

3.52 Asn1UniversalString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Type](#).

Public Member Functions

- [Asn1UniversalString](#) (System.String value)
- [Asn1UniversalString](#) (int[] value)
- [Asn1UniversalString](#) ()
- virtual void [Decode](#) (Asn1JsonDecodeBuffer buffer)
- void [Decode](#) (Asn1OerDecodeBuffer buffer, int length)
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1JsonOutputStream outs)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1OerEncodeBuffer buffer, bool withLength)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- virtual void [EncodeData](#) (Asn1XmlXerEncoder buffer)
- override bool [Equals](#) (System.Object value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()
- bool [validate](#) ([Asn1CharSet](#) charSet)

Public Attributes

- const int [BITSPERCHAR](#) = 32
- int[] [mValue](#)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Protected Member Functions

- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int nchars, int abpc, int ubpc, [Asn1CharSet](#) charSet, int startIdx)
- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerOutputStream outs, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Encode](#) (Asn1PerOutputStream outs, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerOutputStream outs, int nchars, int offset, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int nchars, int offset, int abpc, int ubpc, [Asn1CharSet](#) charSet)

Protected Attributes

- internal System.Text.StringBuilder [mStringBuffer](#)

Properties

- override int [Length](#) [get]

3.52.1 Detailed Description

This is a container class for holding the components of an ASN.1 Universal string value.

3.52.2 Constructor & Destructor Documentation

3.52.2.1 [Asn1UniversalString \(\)](#)

This constructor creates an empty string that can be used in a Decode method call to receive a string value.

3.52.2.2 [Asn1UniversalString \(int\[\] value\)](#)

This constructor initializes the universal string from the given an array of 32-bit integer value.

Parameters

value universal string value as array of 32-bit int

3.52.2.3 [Asn1UniversalString \(System.String value\)](#)

This constructor converts a standard C# string value into a universal string.

Parameters

value universal string value as string

3.52.3 Member Function Documentation

3.52.3.1 virtual void Decode (Asn1JsonDecodeBuffer *buffer*) [virtual]

Decode ASN.1 restricted character string from JSON.

3.52.3.2 void Decode (Asn1OerDecodeBuffer *buffer*, int *length*)

Decode the value in accordance with OER.

This class's implementation decodes a string of the given length. This method is final as I don't see any reason for overriding it.

Parameters

length Length of string to decode, in characters.

3.52.3.3 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode the value in accordance with OER.

This class's implementation decodes the string with a length determinant. If a subclass should be decoded without a length determinant, it should override this method to invoke Decode(*buffer*, *length*). Subclasses may override this to add constraint checks after invoking this method to decode the string.

Reimplemented from [Asn1Type](#).

3.52.3.4 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]

This overloaded version of the Decode method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable.

Parameters

buffer Decode message buffer object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.52.3.5 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *nchars*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, int *startIndex*) [protected, virtual]

This method decodes the contents of a UniversalString. This version of the method assumes a permitted alphabet constraint is in place.

Parameters

- buffer* Decode message buffer object
- nchars* Number of characters
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)
- startIdx* Start index to fill in value array

3.52.3.6 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [**protected**, **virtual**]

This method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes that a permitted alphabet constraint has been specified that would reduce the number of bits-per-character from the default character set. It also assumes a general length determinant is present (i.e. there is not size constraint). The decoded result is stored in the public `mValue` member variable.

Parameters

- buffer* Decode message buffer object
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)

3.52.3.7 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*) [**virtual**]

This overloaded version of the Decode method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable.

Parameters

- buffer* Decode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

3.52.3.8 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*) [**virtual**]

This method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but allows a permitted alphabet character set to be specified. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable.

Parameters

- buffer* Decode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)

3.52.3.9 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.52.3.10 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 universal string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.52.3.11 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 Universal String value using the XML encoding rules (XER).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.52.3.12 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 Universal String value using the XML schema encoding rules(asn2xsd).

Parameters

buffer String containing data to be decoded

attrs Attributes string from element tag

3.52.3.13 virtual internal void Encode (Asn1PerOutputStream *outs*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing the permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

Exceptions

Asn1Exception Thrown, if operation is failed.

3.52.3.14 virtual internal void Encode (Asn1PerOutputStream outs, int abpc, int ubpc, Asn1CharSet charSet) [protected, virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing the permitted alphabet constraint character set (optional)

Exceptions

Asn1Exception Thrown, if operation is failed.

3.52.3.15 virtual internal void Encode (Asn1PerOutputStream outs, int nchars, int offset, int abpc, int ubpc, Asn1CharSet charSet) [protected, virtual]

This method encodes the contents of a UniversalString type. This version assumes a permitted alphabet constraint was specified.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

nchars Number of characters from string to encode

offset Offset to first char in string to encode

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing permitted alphabet constraint character set (optional)

Exceptions

Asn1Exception Thrown, if operation is failed.

3.52.3.16 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet, long lower, long upper) [virtual]

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

charSet Object representing the permitted alphabet constraint character set

lower Effective size constraint lower bound

upper Effective size constraint upper bound

Exceptions

Asn1Exception Thrown, if operation is failed.

3.52.3.17 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet) [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

charSet Object representing permitted alphabet constraint character set (optional)

Exceptions

Asn1Exception Thrown, if operation is failed.

3.52.3.18 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Output Stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.52.3.19 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 universal string including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.52.3.20 virtual void Encode (Asn1JsonOutputStream outs) [virtual]

Encode the value of this object as a JSON string.

Parameters

out

3.52.3.21 override void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix) [virtual]

This method encodes an ASN.1 Universal String value with element and namespace prefix name tag using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

elemName Element name

nsPrefix Element namespace value

Reimplemented from [Asn1Type](#).

3.52.3.22 **override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]**

This method encodes an ASN.1 Universal String value using the XML encoding rules (XER).

Parameters

buffer Encode message buffer object

elemName Element name

Reimplemented from [Asn1Type](#).

3.52.3.23 **virtual void Encode (Asn1OerEncodeBuffer *buffer*, bool *withLength*) [virtual]**

Encode the string, with or without a length determinant.

Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters

withLength true if a length determinant should be encoded.

3.52.3.24 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

Encode the value in accordance with OER.

This class's implementation invokes encode(buffer, true) to encode the string with a length determinant. If a subclass should be encoded without a length determinant, it should override this to invoke Encode(buffer, false).

Reimplemented from [Asn1Type](#).

3.52.3.25 **virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]**

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing the permitted alphabet constraint character set (optional)

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.52.3.26 virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [protected, virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing the permitted alphabet constraint character set (optional)

3.52.3.27 virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *nchars*, int *offset*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [protected, virtual]

This method encodes the contents of a UniversalString type. This version assumes a permitted alphabet constraint was specified.

Parameters

buffer Encode message buffer object

nchars Number of characters from string to encode

offset Offset to first char in string to encode

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing permitted alphabet constraint character set (optional)

3.52.3.28 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

charSet Object representing the permitted alphabet constraint character set

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.52.3.29 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*) [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

charSet Object representing permitted alphabet constraint character set (optional)

3.52.3.30 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

3.52.3.31 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 universal string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.52.3.32 virtual void EncodeData (Asn1XmlXerEncoder *buffer*) [virtual]

This method encodes an ASN.1 Universal String value using the XML Encoding as specified in the XML schema standard(asn2xsd).

Parameters

buffer Encode message buffer object

3.52.3.33 **override bool Equals (System.Object value)**

This method compares this character string value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [Asn1UniversalString](#).

Returns

`true` if the specified Object is equal to the current Object; otherwise, `false`.

3.52.3.34 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.52.3.35 **override System.String ToString ()**

This method will return a string representation of the value. The format is the ASN.1 value format for this type.

Returns

Stringified representation of the value

3.52.3.36 **bool validate (Asn1CharSet charSet)**

This method will attempt to validate a string against its internal character set.

Returns

True or False.

3.52.4 **Member Data Documentation**

3.52.4.1 **new readonly Asn1Tag _TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 28).

Reimplemented from [Asn1Type](#).

3.52.4.2 **const int BITSPERCHAR = 32**

The `BITSPERCHAR` constant specifies the number of bits per character for PER (aligned or unaligned).

3.52.4.3 **internal System.Text.StringBuilder mStringBuffer [protected]**

Variable holds the string representation of the value

3.52.4.4 `int [] mValue`

The `mValue` public member variable is used to hold the string value to be encoded or the results of a Decode operation. For `UniversalString`, the characters are stored in an array of 32-bit integer values.

3.52.5 Property Documentation

3.52.5.1 `override int Length [get]`

Gets the length of the character string in characters.

Value: number of characters.

Reimplemented from [Asn1Type](#).

3.53 Asn1UTCTime Class Reference

Public Member Functions

- [Asn1UTCTime](#) (System.String data, bool useDerRules)
- [Asn1UTCTime](#) (System.String data)
- [Asn1UTCTime](#) (bool useDerRules)
- [Asn1UTCTime](#) ()
- override void [Clear](#) ()
- override System.Int32 [CompareTo](#) (System.Object obj)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [ParseString](#) (System.String data)
- override void [SetTime](#) (System.DateTime time)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Protected Member Functions

- internal override bool [CompileString](#) ()
- internal override void [Init](#) ()

Properties

- override string [Fraction](#) [get, set]
- override int [Year](#) [get, set]

3.53.1 Detailed Description

This is a container class for holding the components of an ASN.1 UTC time string value.

3.53.2 Constructor & Destructor Documentation

3.53.2.1 Asn1UTCTime ()

The default constructor creates an empty time string object.

3.53.2.2 Asn1UTCTime (bool useDerRules)

This constructor creates an empty UTCTime string object and allows DER encoding rules to be specified.

Parameters

useDerRules 'true' if time string should be encoded with DER/PER.

3.53.2.3 `Asn1UTCTime (System.String data)`

This version of the constructor can be used to set the string `mValue` member variable to the given time string.

Parameters

data UTCTime as string

3.53.2.4 `Asn1UTCTime (System.String data, bool useDerRules)`

This version of the constructor can be used to set the string `mValue` member variable to the given time string and specify DER encoding rules be used to construct the string.

Parameters

data UTCTime as string

useDerRules 'true' if time string should be encoded with DER/PER.

3.53.3 Member Function Documentation

3.53.3.1 `override void Clear ()`

Clears out time string.

3.53.3.2 `override System.Int32 CompareTo (System.Object obj)`

This method compares this object with [Asn1Time](#) class instance or with `System.DateTime` instance. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

Parameters

obj the Object to be compared.

Returns

The difference in Ticks with the specified object.

3.53.3.3 `internal override bool CompileString () [protected]`

Compiles new time string according X.680 (clause 42) and ISO 8601.

Returns

`true` if successful; otherwise `false`.

3.53.3.4 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*)

This method decodes an ASN.1 UTCTime value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

- buffer* Decode message buffer object
- explicitTagging* Flag indicating element is explicitly tagged
- implicitLength* Length of contents if implicit

3.53.3.5 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*)

This method encodes and writes to the stream an ASN.1 UTC time string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

- outs* BER Output Stream object
- explicitTagging* Flag indicating explicit tagging should be done

Exceptions

- [Asn1Exception](#) Thrown, if operation is failed.

3.53.3.6 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*)

This method encodes an ASN.1 UTCTime type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

- buffer* Encode message buffer object
- explicitTagging* Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

3.53.3.7 internal override void Init () [protected]

This method initializes the [Asn1UTCTime](#) class member variables.

3.53.3.8 override void ParseString (System.String *data*)

This method parses passed UTCTime string.

Parameters

- data* The UTCTime string value to be parsed.

Exceptions

Asn1Exception Thrown, if operation is failed.

3.53.3.9 override void SetTime (System.DateTime time)

This method converts the System.DateTime value to UTCTime string.

Parameters

time The System.DateTime value.

Exceptions

Asn1Exception Thrown, if operation is failed.

3.53.4 Member Data Documentation

3.53.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 23).

3.53.5 Property Documentation

3.53.5.1 override string Fraction [get, set]

Gets the Fraction component of the UTC Time. Which is always Zero(i.e. empty string)

Value: Zero or empty string

Exceptions

Asn1Exception Always thrown for Sets call.

3.53.5.2 override int Year [get, set]

Gets or Sets the year component of the time value. You may pass 'year' parameter either as two last digits of the year (00 - 99) or as full 4 digits (0 - 9999). Note Gets returns year in full 4 digits format.

Value: Year component (full 4 digits).

Exceptions

Asn1Exception Thrown, if operation is failed.

3.54 Asn1UTF8String Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1CharString](#).

Inherited by [Asn1Real10](#).

Public Member Functions

- [Asn1UTF8String](#) (System.String data)
- [Asn1UTF8String](#) ()
- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- void [SetAnyAttribute](#) (String qname, String val)

Static Public Member Functions

- static string [Decode](#) (Asn1BerDecodeBuffer buffer, [Asn1Tag](#) explicitTag, int implicitLength)
- static String [DecodeUTF8](#) (Asn1OerDecodeBuffer buffer)
- static string [DecodeUTF8](#) (Asn1PerDecodeBuffer buffer)
- static void [Encode](#) (Asn1BerOutputStream outs, [Asn1Tag](#) explicitTag, string value)
- static void [Encode](#) (Asn1OerEncodeBuffer buffer, String value)
- static void [Encode](#) (Asn1PerEncodeBuffer buffer, string value)
- static int [Encode](#) (Asn1BerEncodeBuffer buffer, [Asn1Tag](#) explicitTag, string value)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

Protected Member Functions

- virtual void [Decode](#) (Asn1OerDecodeBuffer buffer, int length)

3.54.1 Detailed Description

This is a container class for holding the components of an ASN.1 UTF-8 string value.

3.54.2 Constructor & Destructor Documentation

3.54.2.1 Asn1UTF8String ()

The default constructor creates an empty time string object.

3.54.2.2 Asn1UTF8String (System.String data)

This constructor can be used to set the UTF8 String `mValue` member variable to the given string value.

Parameters

data UTF8 String value

3.54.3 Member Function Documentation

3.54.3.1 virtual void Decode (Asn1OerDecodeBuffer buffer, int length) [protected, virtual]

This method decodes the content of an ASN.1 UTF8String string value that was encoded according to OER. The length is not decoded but is taken to be as given.

Parameters

buffer Decode message buffer object

3.54.3.2 override void Decode (Asn1OerDecodeBuffer buffer) [virtual]

This method decodes an ASN.1 UTF8String string value that was encoded according to OER.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.3 virtual void Decode (Asn1PerDecodeBuffer buffer, long lower, long upper) [virtual]

This method decodes a sized ASN.1 UTF-8 string value using the packed encoding rules (PER).

Parameters

buffer Decode message buffer object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

3.54.3.4 override void Decode (Asn1PerDecodeBuffer buffer) [virtual]

This method decodes an ASN.1 UTF-8 string value using the packed encoding rules (PER). The string is assumed to not contain a size constraint.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.5 `override void Decode (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)`
[virtual]

This method decodes an ASN.1 UTF-8 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This string type uses variable length character encodings.

Parameters

buffer Decode message buffer object
explicitTagging Flag indicating element is explicitly tagged
implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.6 `static string Decode (Asn1BerDecodeBuffer buffer, Asn1Tag explicitTag, int implicitLength)`
[static]

This method decodes an ASN.1 UTF-8 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This string type uses variable length character encodings.

BER encodes UTF8String, OID-IRI, and RELATIVE-OID-IRI in essentially the same way; this permits sharing of the implementation.

Parameters

buffer Decode message buffer object
explicitTag Tag to explicitly match, or null if none
implicitLength Length of contents if implicit

3.54.3.7 `static String DecodeUTF8 (Asn1OerDecodeBuffer buffer)` **[static]**

This method decodes the content of an ASN.1 UTF8String string value that was encoded according to OER.

Parameters

buffer Decode message buffer object

Returns

The decoded string.

3.54.3.8 `static string DecodeUTF8 (Asn1PerDecodeBuffer buffer)` **[static]**

This method decodes an ASN.1 UTF-8 string value using the packed encoding rules (PER).

Parameters

buffer Decode message buffer object

3.54.3.9 virtual void Encode (Asn1PerOutputStream outs, long lower, long upper) [virtual]

This method encodes a size-constrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Also throws any exception thrown by the Asn1PerOutputStream.

Parameters

outs PER Output Stream object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.54.3.10 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an unconstrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Also throws any exception thrown by the Asn1PerOutputStream.

Parameters

outs PER Output Stream object

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.11 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 UTF8 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.12 **static void Encode (Asn1BerOutputStream *outs*, Asn1Tag *explicitTag*, string *value*) [static]**

This method encodes the given string using the BER encoding rules for UTF8String, including the given tag, if provided.

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTag Tag to encode, or null for none.

value The value to encode. For OID-IRI and RELATIVE-OID-IRI, should not contain whitespace.

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.54.3.13 **static void Encode (Asn1OerEncodeBuffer *buffer*, String *value*) [static]**

This method encodes an ASN.1 UTF8String according to OER.

Parameters

buffer Encode message buffer object

value The value to be encoded.

3.54.3.14 **override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]**

This method encodes a ASN.1 UTF8String according to OER.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.15 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]**

This method encodes a size-constrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Parameters

buffer Encode message buffer object

lower Lower bound (inclusive) of size constraint

upper Upper bound (inclusive) of size constraint

3.54.3.16 **override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an unconstrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Parameters

buffer Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.17 **static void Encode (Asn1PerEncodeBuffer *buffer*, string *value*) [static]**

This method encodes a string using the packed encoding rules (PER) specific for ASN1. UTF8String. These rules are essentially shared with OID-IRI and RELATIVE-OID-IRI.

Parameters

buffer Encode message buffer object

value The value to be encoded. In the case of OID-IRI and RELATIVE-OID-IRI, should not contain whitespace.

3.54.3.18 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]**

This method encodes an ASN.1 UTF8 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.54.3.19 **static int Encode (Asn1BerEncodeBuffer *buffer*, Asn1Tag *explicitTag*, string *value*) [static]**

This method encodes an ASN.1 UTF8 string type. Nearly the same encoding is shared by OID-IRI and RELATIVE-OID_IRI; this method facilitates sharing the implementation.

The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTag Tag to explicitly encode, or null for none.

value The value to encode. For OID-IRI and RELATIVE-OID-IRI, whitespace should already have been removed.

Returns

Length in octets of encoded component

3.54.3.20 void SetAnyAttribute (String *qname*, String *val*)

This method will set the anyAttribute type value for given qname and value of XML attribute

Parameters

qname The qualified (prefixed) name

value The attribute value

3.54.4 Member Data Documentation

3.54.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 12).

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1Real10](#).

3.55 Asn1Util Class Reference

Static Public Member Functions

- static System.String [BCDToString](#) (byte[] bcd)
- static byte[] [DecodeBase64Array](#) (char[] srcArray)
- static string [EncodeBase64Array](#) (byte[] srcArray)
- static byte[] [GetAddressBytes](#) (string ipAddress)
- static int [GetBytesCount](#) (long val)
- static int [GetUlongBytesCount](#) (long val)
- static bool [IsLimited](#) ()
- static byte[] [StringToBCD](#) (System.String str)
- static byte[] [StringToTBCD](#) (System.String str)
- static String [StripWhitespace](#) (String value)
- static System.String [TBCDToString](#) (byte[] bcd)
- static void [ToArray](#) (System.Collections.ICollection c, System.Object[] objects)
- static byte[] [ToByteArray](#) (System.String sourceString)
- static char[] [ToCharArray](#) (byte[] byteArray)
- static System.String [ToHexString](#) (byte[] b, int offset, int nbytes, bool spaces)
- static System.String [ToHexString](#) (byte[] b, int offset, int nbytes)
- static System.String [ToHexString](#) (byte b)
- static long [URShift](#) (long number, long bits)
- static long [URShift](#) (long number, int bits)
- static int [URShift](#) (int number, long bits)
- static int [URShift](#) (int number, int bits)
- static void [WriteStackTrace](#) (System.Exception throwable, System.IO.TextWriter stream)

3.55.1 Detailed Description

This class contains some general purpose static utility functions.

3.55.2 Member Function Documentation

3.55.2.1 static System.String [BCDToString](#) (byte[] *bcd*) [static]

Translates a BCD string to an ASCII string. The 0xF half-byte is assumed as end of string indicator.

Parameters

bcd the source BCD string

Returns

the ASCII string.

3.55.2.2 `static byte [] DecodeBase64Array (char[] srcArray) [static]`

Translates the specified Base64 array into byte array. The resulting array could be converted to the String by new String (byte[]) constructor.

Parameters

srcArray Base64 byte array to be translated

Returns

decoded byte array

3.55.2.3 `static string EncodeBase64Array (byte[] srcArray) [static]`

Translates the specified byte array to Base64 string.

Parameters

srcArray byte array to be translated

Returns

Base64 encoded string

3.55.2.4 `static byte [] GetAddressBytes (string ipaddress) [static]`

Converts an IPAddress to byte array

Parameters

ipaddress String representation of IP Address

Returns

The byte array(size 4) representation of IP address

3.55.2.5 `static int GetBytesCount (long val) [static]`

Calculate the number of bytes necessary to represent a signed long value.

Parameters

val signed long value.

Returns

the number of bytes.

3.55.2.6 `static int GetUlongBytesCount (long val) [static]`

Calculate the number of bytes necessary to represent an unsigned long value.

Parameters

val unsigned long value.

Returns

the number of bytes.

3.55.2.7 `static bool IsLimited () [static]`

Indicates whether the run-time is limited or unlimited.

Returns

true if limited, false if unlimited

3.55.2.8 `static byte [] StringToBCD (System.String str) [static]`

Translates an ASCII string to a BCD string. The ASCII string must contain only characters in the range [0..9] & ([A..F] | [a..f]). If the length of the source string is not even, the unused part of the last byte will be set to 0xF.

Parameters

str the source ASCII string

Returns

the BCD string as a byte array.

Exceptions

[*AsnIValueParseException*](#) If invalid characters are in the source string.

3.55.2.9 `static byte [] StringToTBCD (System.String str) [static]`

Translates an ASCII string to a TBCD string. The ASCII string must contain only characters in the range [0..9] & ([A..F] | [a..f]). If the length of the source string is not even, the unused part of the last byte will be set to 0xF. TBCD strings differ from BCD strings in that the least significant nibble is set in the upper four bits; so the string "12345" would be translated "0x2143f5".

Parameters

str the source ASCII string

Returns

the TBCD string as a byte array.

Exceptions

[*AsnIValueParseException*](#) If invalid characters are in the source string.

3.55.2.10 `static String StripWhitespace (String value) [static]`

Return the given string with all whitespace characters removed. Here, "whitespace characters" means those characters defined by X.680 12.1.6 as whitespace: HT, LF, VT, FF, CR, SPACE.

Returns

value, with all whitespace removed; if the input string has no whitespace, it is returned itself.

3.55.2.11 `static System.String TBCDToString (byte[] bcd) [static]`

Translates a TBCD string to an ASCII string. The 0xF half-byte is assumed as end of string indicator.

Parameters

bcd the source TBCD string

Returns

the ASCII string.

3.55.2.12 `static void ToArray (System.Collections.ICollection c, System.Object[] objects) [static]`

Obtains an array containing all the elements of the collection.

Parameters

c The Collection instance, which contains the elements.

objects The array into which the elements of the collection will be stored.

Returns

The array containing all the elements of the collection.

3.55.2.13 `static byte [] ToByteArray (System.String sourceString) [static]`

Converts a string to an array of bytes

Parameters

sourceString The string to be converted

Returns

The new array of bytes

3.55.2.14 `static char [] ToCharArray (byte[] byteArray) [static]`

Converts an array of bytes to an array of chars

Parameters

byteArray The array of bytes to convert

Returns

The new array of chars

3.55.2.15 `static System.String ToHexString (byte[] b, int offset, int nbytes, bool spaces) [static]`

Convert a array of bytes into a hex string.

Parameters

b byte array to be converted to hex string

offset start position in byte array

nbytes no. of bytes to be converted

spaces Pass true if each byte's hex digits should be followed by a space character.

Returns

Hex String value

3.55.2.16 `static System.String ToHexString (byte[] b, int offset, int nbytes) [static]`

Convert a array of bytes into a hex string.

Parameters

b byte array to be converted to hex string

offset start position in byte array

nbytes no. of bytes to be converted

Returns

Hex String value

3.55.2.17 `static System.String ToHexString (byte b) [static]`

Convert a byte value to a hex string. Unlike the C# built-in function, this will:

a. not sign extend the byte value out to 32 bits if the MSB is set, and b. put a zero padding byte in front if less than 0xf

In other words, a character string of length 2 is always returned.

Parameters

b byte value

Returns

Hex String value

3.55.2.18 **static long URShift (long number, long bits) [static]**

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

Parameters

number Number to operate on

bits Amount of bits to shift

Returns

The resulting number from the shift operation

3.55.2.19 **static long URShift (long number, int bits) [static]**

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

Parameters

number Number to operate on

bits Amount of bits to shift

Returns

The resulting number from the shift operation

3.55.2.20 **static int URShift (int number, long bits) [static]**

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

Parameters

number Number to operate on

bits Amount of bits to shift

Returns

The resulting number from the shift operation

3.55.2.21 **static int URShift (int number, int bits) [static]**

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

Parameters

number Number to operate on

bits Amount of bits to shift

Returns

The resulting number from the shift operation

3.55.2.22 `static void WriteStackTrace (System.Exception throwable, System.IO.TextWriter stream)`
`[static]`

Writes the exception stack trace to the received stream

Parameters

throwable Exception to obtain information from

stream Output stream used to write to

3.56 Asn1Value Class Reference

Static Public Member Functions

- static byte[] [ParseString](#) (System.String data)
- static byte[] [ParseString](#) (System.String data, [IntHolder](#) numbits)
- static bool [StringEqualsBytes](#) (String value, byte[] data, int nbits)
- static bool [StringEqualsBytes](#) (String value, byte[] data)

3.56.1 Detailed Description

This class provides methods for parsing and formatting text in ASN.1 value notation.

3.56.2 Member Function Documentation

3.56.2.1 static byte [] ParseString (System.String data) [static]

This overloaded version of the ParseString method sets the numbits holder value to null.

Parameters

data The ASN.1 value specification text

Returns

byte array value

3.56.2.2 static byte [] ParseString (System.String data, IntHolder numbits) [static]

This static method parses the given ASN.1 value text (either a binary or hex data string) and returns the value in a binary byte array. The number of bits is also returned.

Examples of valid value formats are as follows:

Binary string: '11010010111001'B

Hex string: '0fa56920014abc'H

Char string: 'abcdefg'

Parameters

data The ASN.1 value specification text

numbits Holder to receive number of bits in string

Returns

byte array value

3.56.2.3 `static bool StringEqualsBytes (String value, byte[] data, int nbits) [static]`

This static method parses the given ASN.1 value text (either a binary or hex data string) and compares it with the given byte array. This version of the method allows you to specify a number of bits to compare, which is useful when dealing with bit strings, as opposed to octet strings.

Examples of valid value formats are as follows: Binary string: '11010010111001'B Hex string: '0fa56920014abc'H Char string: 'abcdefg' /p>

Parameters

value The ASN.1 value specification text

data Array of bytes to compare string with.

nbits Number of bits in data to compare.

3.56.2.4 `static bool StringEqualsBytes (String value, byte[] data) [static]`

This static method parses the given ASN.1 value text (either a binary or hex data string) and compares it with the given byte array. Examples of valid value formats are as follows: Binary string: '11010010111001'B Hex string: '0fa56920014abc'H Char string: 'abcdefg'

Parameters

value The ASN.1 value specification text

data Array of bytes to compare string with.

3.57 Asn1ValueParseException Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1Exception](#).

Public Member Functions

- [Asn1ValueParseException](#) (System.String text, int offset)
- [Asn1ValueParseException](#) (System.String text)

3.57.1 Detailed Description

This class defines the 'ASN.1 value Parse' exception that is thrown when a string containing an ASN.1 value cannot be parsed.

3.57.2 Constructor & Destructor Documentation

3.57.2.1 Asn1ValueParseException (System.String text)

This constructor creates an exception object with a textual message describing the expected and parsed tag values.

Parameters

text The value string that could not be parsed

3.57.2.2 Asn1ValueParseException (System.String text, int offset)

This constructor creates an exception object with a textual message describing the expected and parsed tag values. This version allows the offset in the string to also be specified.

Parameters

text The value string that could not be parsed

offset Offset to error location in string

3.58 Asn1VarWidthCharString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1CharString](#).

Inherited by [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1ObjectDescriptor](#), [Asn1T61String](#), and [Asn1VideotexString](#).

Public Member Functions

- override void [Decode](#) (Asn1OerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1OerEncodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)

Public Attributes

- const int [BITSPERCHAR_A](#) = 8
- const int [BITSPERCHAR_U](#) = 8

Protected Member Functions

- internal [Asn1VarWidthCharString](#) (System.String data, short typeCode)
- internal [Asn1VarWidthCharString](#) (short typeCode)

3.58.1 Detailed Description

This is an abstract base class for holding the ASN.1 variable width character string types ([GraphicString](#), [GeneralString](#), [TeletexString](#), [T61String](#), [VideotexString](#), [ObjectDescriptor](#)).

3.58.2 Constructor & Destructor Documentation

3.58.2.1 internal Asn1VarWidthCharString (short typeCode) [protected]

The default constructor creates an empty string object.

Parameters

typeCode Universal ID code for ASN.1 character string

3.58.2.2 internal Asn1VarWidthCharString (System.String data, short typeCode) [protected]

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data Character string

typeCode Universal ID code for ASN.1 character string

3.58.3 Member Function Documentation

3.58.3.1 override void Decode (Asn1OerDecodeBuffer *buffer*) [virtual]

Decode the value in accordance with OER.

Subclasses may override this to add constraint checks after invoking this method to decode the string.

Reimplemented from [Asn1Type](#).

3.58.3.2 virtual void Decode (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This overloaded version of the Decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present. A permitted alphabet may be passed, but it will be ignored.

The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Decode message buffer object

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.58.3.3 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Parameters

buffer Decode message buffer object

Reimplemented from [Asn1Type](#).

3.58.3.4 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value directly into the stream, in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present. A permitted alphabet may be passed, but it will be ignored.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Also throws any exception thrown by the [Asn1PerOutputStream](#).

Parameters

outs PER Encode message stream object

lower Effective size constraint lower bound

upper Effective size constraint upper bound

Exceptions

[Asn1Exception](#) Thrown, if operation is failed.

3.58.3.5 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream. This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the `Asn1CharString` base class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

Parameters

outs PER Encode message stream object

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from `Asn1Type`.

3.58.3.6 override void Encode (Asn1OerEncodeBuffer *buffer*) [virtual]

Encode the value in accordance with OER.

We're assuming each char is encoded in a single byte of the same value as the char.

Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Reimplemented from `Asn1Type`.

3.58.3.7 virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present. A permitted alphabet may be passed, but it will be ignored.

The value to encode is stored in the public `mValue` member variable in the `Asn1CharString` base class.

Parameters

buffer Encode message buffer object

lower Effective size constraint lower bound

upper Effective size constraint upper bound

3.58.3.8 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the `Asn1CharString` base class.

Parameters

buffer Encode message buffer object

Reimplemented from `Asn1Type`.

3.58.4 Member Data Documentation

3.58.4.1 const int BITSPERCHAR_A = 8

The BITSPERCHAR_A constant specifies the number of bits per character for PER (aligned).

3.58.4.2 const int BITSPERCHAR_U = 8

The BITSPERCHAR_U constant specifies the number of bits per character for PER (unaligned).

3.59 Asn1VideotexString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString](#).

Public Member Functions

- [Asn1VideotexString](#) (System.String data)
- [Asn1VideotexString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.59.1 Detailed Description

This is a container class for holding the components of an ASN.1 videotex string value.

3.59.2 Constructor & Destructor Documentation

3.59.2.1 Asn1VideotexString ()

The default constructor creates an empty string object.

3.59.2.2 Asn1VideotexString (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

Parameters

data string representation of videotex string

3.59.3 Member Function Documentation

3.59.3.1 override void Decode (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength) [virtual]

This method decodes an ASN.1 Videotex string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.59.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 videotex string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Reimplemented from [Asn1Type](#).

3.59.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 Videotex String type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.59.4 Member Data Documentation

3.59.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 21).

Reimplemented from [Asn1Type](#).

3.60 Asn1VisibleString Class Reference

Inherits [Com::Objsys::Asn1::Runtime::Asn18BitCharString](#).

Public Member Functions

- [Asn1VisibleString](#) (System.String data)
- [Asn1VisibleString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

Static Public Attributes

- static new readonly [Asn1Tag_TAG](#)

3.60.1 Detailed Description

This is a container class for holding the components of an ASN.1 Visible string value.

3.60.2 Constructor & Destructor Documentation

3.60.2.1 Asn1VisibleString ()

The default constructor creates an empty string object.

3.60.2.2 Asn1VisibleString (System.String data)

This version of the constructor can be used to set the Visible string `mValue` member variable to the given string.

Parameters

data string representation of visible string

3.60.3 Member Function Documentation

3.60.3.1 override void Decode (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength) [virtual]

This method decodes an ASN.1 Visible string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Parameters

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

Reimplemented from [Asn1Type](#).

3.60.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 visible string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

Parameters

outs BER Output Stream object

explicitTagging Flag indicating explicit tagging should be done

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

3.60.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 Visible string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters

buffer Encode message buffer object

explicitTagging Flag indicating explicit tagging should be done

Returns

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

3.60.4 Member Data Documentation

3.60.4.1 new readonly Asn1Tag _TAG [static]

The _TAG constant describes the universal tag for this data type (UNIVERSAL 26).

Reimplemented from [Asn1Type](#).

3.61 BigInteger Class Reference

Public Member Functions

- [BigInteger Add](#) ([BigInteger](#) op)
- [BigInteger](#) (System.String value, int radix)
- [BigInteger](#) (System.Int64 value)
- [BigInteger](#) (System.String value)
- [BigInteger](#) (byte[] value, int sign)
- [BigInteger](#) ()
- int [BitLength](#) ()
- virtual int [CompareTo](#) ([BigInteger](#) value)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (long value)
- byte[] [GetData](#) ()
- override int [GetHashCode](#) ()
- void [Init](#) (System.String val, int radix)
- bool [IsNegative](#) ()
- long [LongValue](#) ()
- void [SecureDelete](#) ()
- void [SetData](#) (byte[] ivalue)
- [BigInteger Subtract](#) ([BigInteger](#) op)
- override System.String [ToString](#) ()
- System.String [ToString](#) (int radix)

Static Public Member Functions

- static implicit [operator BigInteger](#) (long value)

3.61.1 Detailed Description

This class represents an ASN.1 INTEGER built-in type. In this case, the values can be greater than 64 bits in size. This class is used in generated source code if the <BigInteger> qualifier is specified in a compiler configuration file.

3.61.2 Constructor & Destructor Documentation

3.61.2.1 BigInteger ()

The default constructor sets the big integer value object reference to null.

3.61.2.2 BigInteger (byte[] value, int sign)

This constructor creates a new big integer object and sets it to the byte[] value passed in.

Parameters

value String value

sign Can be -1 for negative, 0 for zero, or 1 for positive.

3.61.2.3 **BigInteger** (System.String *value*)

This constructor creates a new big integer object and sets it to the string value passed in. String value may contain the prefix that describes the radix: 0x - hexadecimal, 0o - octal, 0b - binary. The string value without prefix assumes decimal value. The optional sign '-' may be specified at the beginning of the string to specify the negative value.

Parameters

value String value

3.61.2.4 **BigInteger** (System.Int64 *value*)

This constructor creates a new big integer object and sets it to the int value passed in.

Parameters

value Integer value

3.61.2.5 **BigInteger** (System.String *value*, int *radix*)

This constructor creates a new big integer object and sets it to the string value passed in. String value may contain the prefix that describes the radix: 0x - hexadecimal, 0o - octal, 0b - binary. The string value without prefix assumes decimal value. The optional sign '-' may be specified at the beginning of the string to specify the negative value.

Parameters

value String value

radix Can be 16 for hexadecimal, 8 for octal, 2 for binary or 10 for decimal

3.61.3 Member Function Documentation

3.61.3.1 **BigInteger** Add (**BigInteger** *op*)

Return the result of adding *op* to this [BigInteger](#). Does not modify this object.

Parameters

op

Returns

3.61.3.2 int **BitLength** ()

Returns the number of bits in the minimal two's-complement representation of this [BigInteger](#), excluding a sign bit. For positive [BigIntegers](#), this is equivalent to the number of bits in the ordinary binary representation. (Computes $\text{ceil}(\log_2(\text{this} < 0 ? -\text{this} : \text{this}+1))$.)

Returns

number of bits in the minimal two's-complement representation of this [BigInteger](#), excluding a sign bit.

3.61.3.3 virtual int CompareTo (BigInteger value) [virtual]

This method compares this integer value to the given value.

Parameters

value The value to compare with the current object.

Returns

-1 if this object is less than value, 0 if this object is equal to value 1 if this object is greater than value

3.61.3.4 override bool Equals (System.Object value)

This method compares this integer value to the given value for equality.

Parameters

value The Object to compare with the current Object. Object should be instance of [BigInteger](#).

Returns

true if the specified Object is equal to the current Object; otherwise, false.

3.61.3.5 virtual bool Equals (long value) [virtual]

This method compares this integer value to the given value for equality.

Parameters

value The long value to compare with the current Object.

Returns

true if the specified long value is equal to the current Object; otherwise, false.

3.61.3.6 byte [] GetData ()

This method provides the byte array representation of the integer value, in 2's complement form. The most significant byte is at index 0.

Returns

byte array for integer value

3.61.3.7 override int GetHashCode ()

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Returns

A hash code for the current Object.

3.61.3.8 void Init (System.String val, int radix)

Translates the String representation of a Integer in the specified radix into a [BigInteger](#). The String representation consists of an optional minus sign followed by a sequence of one or more digits in the specified radix. The String may not contain any extraneous Characters (whitespace, for example).

Parameters

val String representation of Integer.
radix radix to be used in interpreting string value.

Exceptions

System.FormatException *val* is not a valid representation of a [BigInteger](#) in the specified radix, or invalid value of radix.

3.61.3.9 bool IsNegative ()

This method checks the Integer value is negative.

Returns

true if negative; otherwise false

3.61.3.10 long LongValue ()

Converts this [BigInteger](#) to a long. This conversion is analogous to the conversion from long to int: if this [BigInteger](#) is too big to fit in a long, only the low-order 64 bits are returned. Note that this conversion can lose information about the overall magnitude of the [BigInteger](#) value as well as return a result with the opposite sign.

Returns

this [BigInteger](#) converted to a long.

3.61.3.11 static implicit operator BigInteger (long value) [static]

Overloaded implicit conversion operator for long to [BigInteger](#) value

3.61.3.12 void SecureDelete ()

This function clears the current value (by overwriting with zeros). Then sets the value to zero, and passes the old value to garbage collector.

3.61.3.13 void SetData (byte[] ivalue)

This method sets the Integer value from byte array.

Parameters

ivalue byte array of the integer value

Returns

Decoded integer value

See also

<seealso cref=GetData To retrieve Byte Array

3.61.3.14 BigInteger Subtract (BigInteger op)

Subtract given *op* from this value and return the result. This object is not modified.

Parameters

op

Returns**3.61.3.15 override System.String ToString ()**

This method will return a string representation of the integer value. The format is the ASN.1 value format for this type..

Returns

Stringified representation of the value

3.61.3.16 System.String ToString (int radix)

Returns the String representation of this [BigInteger](#) in the given radix. If the radix is invalid, it will default to 10 (as is the case for `Int32.ToString`). The a minus sign is prepended if appropriate. This method is compatible with `BigInteger(String, int)` constructor.

Parameters

radix radix of the String representation.

Returns

String representation of this [BigInteger](#) in the given radix.

See also

<seealso cref=System.Int32.ToString Integer [ToString](#) methods

3.62 BooleanHolder Class Reference

Public Member Functions

- [BooleanHolder](#) (bool value)
- [BooleanHolder](#) ()

Public Attributes

- bool [mValue](#)

3.62.1 Detailed Description

A Holder class for a `Boolean` that is used to store "out" and "inout" parameters in methods. If a method has a `boolean` as an "out" or "inout" parameter, the programmer must pass an instance of [BooleanHolder](#) as the corresponding parameter in the method invocation; for "inout" parameters, the programmer must also fill the "in" value.

If `myBooleanHolder` is an instance of `BooleanHolder`,

the value stored in its value field can be accessed with `myBooleanHolder.mValue`.

3.62.2 Constructor & Destructor Documentation

3.62.2.1 `BooleanHolder` ()

The default constructor for [BooleanHolder](#) class

3.62.2.2 `BooleanHolder` (bool *value*)

This constructor will initialize [BooleanHolder](#) class with specified boolean value.

Parameters

value `Boolean` value

3.62.3 Member Data Documentation

3.62.3.1 `bool mValue`

This member variable is where the boolean value is stored.

3.63 Diag Class Reference

Public Member Functions

- virtual bool [IsEnabled](#) (int traceLevel)
- virtual bool [IsEnabled](#) ()
- virtual void [Println](#) (System.String s, int traceLevel)
- virtual void [Println](#) (System.String s)
- virtual bool [SetEnabled](#) (bool data)
- virtual int [SetTraceLevel2](#) (int level)

Static Public Member Functions

- static void [HexDump](#) (System.IO.Stream istrm, System.IO.StreamWriter ostrm)
- static void [HexDump](#) (byte[] bytes, int traceLevel)
- static void [HexDump](#) (byte[] bytes)
- static [Diag Instance](#) ()
- static void [Prtln](#) (byte[] b, int offset, int nbytes)
- static void [Prtln](#) (byte[] b, int offset, int nbytes, int tl)
- static void [Prtln](#) (System.String s, int traceLevel)
- static void [Prtln](#) (System.String s)
- static int [SetTraceLevel](#) (int level)

Properties

- virtual System.IO.StreamWriter [PrintStream](#) [set]

3.63.1 Detailed Description

This class is used for printing diagnostic messages for debugging the run-time components. It allows messages to be easily switched on and off.

3.63.2 Member Function Documentation

3.63.2.1 static void [HexDump](#) (System.IO.Stream *istrm*, System.IO.StreamWriter *ostrm*) [static]

This method prints a formatted hex dump for the contents of the given input stream to the given output stream.

Parameters

istrm Input Stream containing data to be dumped

ostrm Output Stream to which formatted data is to be written

3.63.2.2 `static void HexDump (byte[] bytes, int traceLevel) [static]`

This method prints a formatted hex dump for the contents of the given byte array to the standard output stream, if the given trace level is enabled.

Parameters

bytes Byte array containing data to be dumped

traceLevel Trace level

3.63.2.3 `static void HexDump (byte[] bytes) [static]`

This method prints a formatted hex dump for the contents of the given byte array to the standard output stream.

Parameters

bytes Byte array containing data to be dumped

3.63.2.4 `static Diag Instance () [static]`

This method provides the current instance of the [Diag](#) Class

Returns

Current instance of the [Diag](#) class

3.63.2.5 `virtual bool IsEnabled (int traceLevel) [virtual]`

This method checks that given trace level message will be printed.

Parameters

traceLevel Trace Level

Returns

true if enabled, else false

3.63.2.6 `virtual bool IsEnabled () [virtual]`

This method will enable the diagnostic message printing.

Returns

true if enabled, else false

3.63.2.7 `virtual void Println (System.String s, int traceLevel) [virtual]`

This method prints a the diagnostic message, if given trace level is enabled

Parameters

s diagnostic message

traceLevel Trace Level

3.63.2.8 virtual void Println (System.String *s*) [virtual]

This method prints a the diagnsotic message

Parameters

s diagnsotic message

3.63.2.9 static void Prtln (byte[] *b*, int *offset*, int *nbytes*) [static]

This method prints a the hex dump of the given byte array to current [Diag](#) class instance

Parameters

b byte array containing data

offset start offset in the byte array

nbytes no of bytes to be printed

3.63.2.10 static void Prtln (byte[] *b*, int *offset*, int *nbytes*, int *tl*) [static]

This method prints a the hex dump of the given byte array to current [Diag](#) class instance, if given trace level is enabled

Parameters

b byte array containing data

offset start offset in the byte array

nbytes no of bytes to be printed

tl trace level

3.63.2.11 static void Prtln (System.String *s*, int *traceLevel*) [static]

This method prints a the diagnsotic message to current [Diag](#) class instance, if given trace level is enabled

Parameters

s diagnsotic message

traceLevel Trace Level

3.63.2.12 static void Prtln (System.String *s*) [static]

This method prints a the diagnsotic message to current [Diag](#) class instance.

Parameters

s diagnsotic message

3.63.2.13 virtual bool SetEnabled (bool *data*) [virtual]

This method enables or disables the diagnostic message printing.

Parameters

data true for enabling printing; otherwise false

Returns

The stat before setting this stat

3.63.2.14 static int SetTraceLevel (int *level*) [static]

This method sets the trace level for the current instance of the [Diag Class](#)

Parameters

level Trace Level

Returns

Set trace level

3.63.2.15 virtual int SetTraceLevel2 (int *level*) [virtual]

This method sets the trace level for this class

Parameters

level Trace Level

Returns

Set trace level

3.63.3 Property Documentation

3.63.3.1 virtual System.IO.StreamWriter PrintStream [set]

Sets the System.IO.StreamWriter object to which the diagnostic messages should be written.

Value: Output stream for diagnostic messages

3.64 IntHolder Class Reference

Public Member Functions

- [IntHolder](#) (int value)
- [IntHolder](#) ()

Public Attributes

- int [mValue](#)

3.64.1 Detailed Description

A Holder class for an `int` that is used to store "out" and "inout" parameters in methods. If a method has an `int` as an "out" or "inout" parameter, the programmer must pass an instance of [IntHolder](#) as the corresponding parameter in the method invocation; for "inout" parameters, the programmer must also fill the "in" value.

If myIntHolder is an instance of IntHolder,

the value stored in its value field can be accessed with `myIntHolder.mValue`.

3.64.2 Constructor & Destructor Documentation

3.64.2.1 IntHolder ()

The default constructor for [IntHolder](#) class

3.64.2.2 IntHolder (int value)

This constructor will initialize [IntHolder](#) class with specified int value.

Parameters

value int value

3.64.3 Member Data Documentation

3.64.3.1 int mValue

This member variable is where the int value is stored.

3.65 SaxHandler Class Reference

Public Member Functions

- override void [Characters](#) (System.Char[] ch, int start, int length)
- override void [EndElement](#) (System.String namespaceURI, System.String localName, System.String qName)
- override void [StartElement](#) (System.String namespaceURI, System.String localName, System.String qName, XmlAttributes atts)

3.65.1 Detailed Description

This class extends the Asn1XerSaxHandler class to add items specific to ASN.1 XER encoding.

3.65.2 Member Function Documentation

3.65.2.1 override void Characters (System.Char[] *ch*, int *start*, int *length*)

This method manage the notification when Characters element were found.

Parameters

- ch* The array with the characters founds
- start* The index of the first position of the characters found
- length* Specify how many characters must be read from the array

3.65.2.2 override void EndElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*)

This method manage the notification when the end element node were found

Parameters

- namespaceURI* The namespace URI of the element
- localName* The local name of the element
- qName* The long name (qualify name) of the element

3.65.2.3 override void StartElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*, XmlAttributes *atts*)

This method manage the event when a start element node were found

Parameters

- namespaceURI* The namespace uri of the element tag
- localName* The local name of the element
- qName* The Qualify (long) name of the element
- atts* The list of attributes of the element

3.66 StringBufferExt Class Reference

Static Public Member Functions

- static `StringBuilder` [Replace](#) (`StringBuilder sbuf`, `int start`, `int end`, `String str`)

3.66.1 Detailed Description

This class provides the additional functionality to `StringBuilder` class

3.66.2 Member Function Documentation

3.66.2.1 static `StringBuilder` `Replace` (`StringBuilder sbuf`, `int start`, `int end`, `String str`) [`static`]

Replaces the characters in a substring of given `StringBuilder` with characters in the specified `String`. The substring begins at the specified `start` and extends to the character at index `end - 1` or to the end of the `StringBuilder` if no such character exists. First the characters in the substring are removed and then the specified `String` is inserted at `start`. (The specified `StringBuilder` will be lengthened to accommodate the specified `String` if necessary.)

Parameters

- sbuf* `StringBuilder` that will have contents.
- start* The beginning index, inclusive.
- end* The ending index, exclusive.
- str* `String` that will replace previous contents.

Returns

The replaced string builder.

3.67 Tokenizer Class Reference

Public Member Functions

- bool [HasMoreTokens](#) ()
- bool [MoveNext](#) ()
- System.String [NextToken](#) (System.String delimiters)
- System.String [NextToken](#) ()
- string [RemainingString](#) ()
- void [Reset](#) ()
- [Tokenizer](#) (System.String source, System.String delimiters, bool includeDelims)
- [Tokenizer](#) (System.String source, System.String delimiters)
- [Tokenizer](#) (System.String source)

Properties

- int [Count](#) [get]
- System.Object [Current](#) [get]

3.67.1 Detailed Description

The class performs token processing in strings

3.67.2 Constructor & Destructor Documentation

3.67.2.1 [Tokenizer \(System.String source\)](#)

Initializes a new class instance with a specified string to process

Parameters

source String to tokenize

3.67.2.2 [Tokenizer \(System.String source, System.String delimiters\)](#)

Initializes a new class instance with a specified string to process and the specified token delimiters to use

Parameters

source String to tokenize

delimiters String containing the delimiters

3.67.2.3 [Tokenizer \(System.String source, System.String delimiters, bool includeDelims\)](#)

Initializes a new class instance with a specified string to process, the specified token delimiters to use, and whether the delimiters must be included in the results.

Parameters

source String to tokenize

delimiters String containing the delimiters

includeDelims Determines if delimiters are included in the results.

3.67.3 Member Function Documentation

3.67.3.1 bool HasMoreTokens ()

Determines if there are more tokens to return from the source string

Returns

True or false, depending if there are more tokens

3.67.3.2 bool MoveNext ()

Performs the same action as HasMoreTokens.

Returns

True or false, depending if there are more tokens

3.67.3.3 System.String NextToken (System.String *delimiters*)

Returns the next token from the source string, using the provided token delimiters

Parameters

delimiters String containing the delimiters to use

Returns

The string value of the token

3.67.3.4 System.String NextToken ()

Returns the next token from the token list

Returns

The string value of the token

3.67.3.5 string RemainingString ()

Returns the rest of the string from current position.

Returns

rest of the string

3.67.3.6 void Reset ()

Does nothing.

3.67.4 Property Documentation

3.67.4.1 int Count [get]

Remaining tokens count

3.67.4.2 System.Object Current [get]

Performs the same action as NextToken.

Index

- [_SetKey](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Type, 209](#)
- [_SetKey2](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Type, 209](#)
- [_TAG](#)
 - [Com::Objsys::Asn1::Runtime::Asn1BigInteger, 21](#)
 - [Com::Objsys::Asn1::Runtime::Asn1BitString, 32](#)
 - [Com::Objsys::Asn1::Runtime::Asn1BMPString, 39](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Boolean, 46](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Enumerated, 86](#)
 - [Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, 91](#)
 - [Com::Objsys::Asn1::Runtime::Asn1GeneralString, 94](#)
 - [Com::Objsys::Asn1::Runtime::Asn1GraphicString, 96](#)
 - [Com::Objsys::Asn1::Runtime::Asn1IA5String, 98](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Integer, 112](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Null, 126](#)
 - [Com::Objsys::Asn1::Runtime::Asn1NumericString, 130](#)
 - [Com::Objsys::Asn1::Runtime::Asn1ObjectDescriptor, 132](#)
 - [Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 138](#)
 - [Com::Objsys::Asn1::Runtime::Asn1OctetString, 149](#)
 - [Com::Objsys::Asn1::Runtime::Asn1PrintableString, 174](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Real, 182](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Real10, 188](#)
 - [Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 193](#)
 - [Com::Objsys::Asn1::Runtime::Asn1T61String, 197](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Type, 220](#)
 - [Com::Objsys::Asn1::Runtime::Asn1UniversalString, 241](#)
 - [Com::Objsys::Asn1::Runtime::Asn1UTCTime, 246](#)
 - [Com::Objsys::Asn1::Runtime::Asn1UTF8String, 253](#)
 - [Com::Objsys::Asn1::Runtime::Asn1VideotexString, 269](#)
 - [Com::Objsys::Asn1::Runtime::Asn1VisibleString, 271](#)
- [Add](#)
 - [Com::Objsys::Asn1::Runtime::BigInteger, 273](#)
- [AddCaptureBuffer](#)
 - [Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 68](#)
- [AddNamedEventHandler](#)
 - [Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, 118](#)
- [Append](#)
 - [Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 134](#)
- [APPL](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Tag, 200](#)
- [Asn18BitCharString](#)
 - [Com::Objsys::Asn1::Runtime::Asn18BitCharString, 8](#)
- [Asn1BigInteger](#)
 - [Com::Objsys::Asn1::Runtime::Asn1BigInteger, 14](#)
- [Asn1BitString](#)
 - [Com::Objsys::Asn1::Runtime::Asn1BitString, 23, 24](#)
- [Asn1BMPString](#)
 - [Com::Objsys::Asn1::Runtime::Asn1BMPString, 34](#)
- [Asn1Boolean](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Boolean, 41](#)
- [Asn1CharRange](#)
 - [Com::Objsys::Asn1::Runtime::Asn1CharRange, 47](#)
- [Asn1CharSet](#)
 - [Com::Objsys::Asn1::Runtime::Asn1CharSet, 50](#)
- [Asn1CharString](#)
 - [Com::Objsys::Asn1::Runtime::Asn1CharString, 54](#)
- [Asn1Choice](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Choice, 60](#)
- [Asn1ConsVioException](#)
 - [Com::Objsys::Asn1::Runtime::Asn1ConsVioException, 66](#)
- [Asn1DiscreteCharSet](#)
 - [Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, 73](#)
- [Asn1EndOfBufferException](#)
 - [Com::Objsys::Asn1::Runtime::Asn1EndOfBufferException, 81](#)
- [Asn1Enumerated](#)
 - [Com::Objsys::Asn1::Runtime::Asn1Enumerated, 83](#)
- [Asn1Exception](#)

Com::Objsys::Asn1::Runtime::Asn1Exception, 88
 Asn1GeneralizedTime
 Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, 89, 90
 Asn1GeneralString
 Com::Objsys::Asn1::Runtime::Asn1GeneralString, 93
 Asn1GraphicString
 Com::Objsys::Asn1::Runtime::Asn1GraphicString, 95
 Asn1IA5String
 Com::Objsys::Asn1::Runtime::Asn1IA5String, 97
 Asn1Integer
 Com::Objsys::Asn1::Runtime::Asn1Integer, 102
 Asn1InvalidArgException
 Com::Objsys::Asn1::Runtime::Asn1InvalidArgException, 113
 Asn1InvalidChoiceOptionException
 Com::Objsys::Asn1::Runtime::Asn1InvalidChoiceOptionException, 114
 Asn1InvalidEnumException
 Com::Objsys::Asn1::Runtime::Asn1InvalidEnumException, 115
 Asn1InvalidLengthException
 Com::Objsys::Asn1::Runtime::Asn1InvalidLengthException, 116
 Asn1InvalidObjectIDException
 Com::Objsys::Asn1::Runtime::Asn1InvalidObjectIDException, 117
 Asn1MissingRequiredException
 Com::Objsys::Asn1::Runtime::Asn1MissingRequiredException, 120
 Asn1NumericString
 Com::Objsys::Asn1::Runtime::Asn1NumericString, 128
 Asn1ObjectDescriptor
 Com::Objsys::Asn1::Runtime::Asn1ObjectDescriptor, 131
 Asn1ObjectIdentifier
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 134
 Asn1OctetString
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 141
 Asn1OpenType
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 158, 159
 Asn1OutputStream
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 167
 Asn1PrintableString
 Com::Objsys::Asn1::Runtime::Asn1PrintableString, 173
 Asn1Real
 Com::Objsys::Asn1::Runtime::Asn1Real, 176
 Asn1Real10
 Com::Objsys::Asn1::Runtime::Asn1Real10, 183
 Asn1RelativeOID
 Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 189
 Asn1SeqOrderException
 Com::Objsys::Asn1::Runtime::Asn1SeqOrderException, 194
 Asn1T61String
 Com::Objsys::Asn1::Runtime::Asn1T61String, 196
 Asn1Tag
 Com::Objsys::Asn1::Runtime::Asn1Tag, 199
 Asn1Time
 Com::Objsys::Asn1::Runtime::Asn1Time, 203
 Asn1TraceHandler
 Com::Objsys::Asn1::Runtime::Asn1TraceHandler, 205
 Asn1UniversalString
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 231
 Asn1UTCTime
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 243, 244
 Asn1UTF8String
 Com::Objsys::Asn1::Runtime::Asn1UTF8String, 247
 Asn1ValueParseException
 Com::Objsys::Asn1::Runtime::Asn1ValueParseException, 263
 Asn1VarWidthCharString
 Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString, 264
 Asn1VideotexString
 Com::Objsys::Asn1::Runtime::Asn1VideotexString, 268
 Asn1VisibleString
 Com::Objsys::Asn1::Runtime::Asn1VisibleString, 270
 Asn1TypeName
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 156
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 166
 Com::Objsys::Asn1::Runtime::Asn1Type, 223
 Available
 Com::Objsys::Asn1::Runtime::Asn1InputStream, 99
 BCDToString
 Com::Objsys::Asn1::Runtime::Asn1Util, 254
 BigInteger
 Com::Objsys::Asn1::Runtime::BigInteger, 272, 273
 BinDump
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 77

Bit8Mask
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200
 BIT_STRING
 Com::Objsys::Asn1::Runtime::Asn1Type, 220
 BitLength
 Com::Objsys::Asn1::Runtime::BigInteger, 273
 BITSPERCHAR
 Com::Objsys::Asn1::Runtime::Asn1BMPString, 39
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 241
 BITSPERCHAR_A
 Com::Objsys::Asn1::Runtime::Asn18BitCharString, 11
 Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString, 267
 BITSPERCHAR_U
 Com::Objsys::Asn1::Runtime::Asn18BitCharString, 11
 Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString, 267
 BMPString
 Com::Objsys::Asn1::Runtime::Asn1Type, 220
 BOOLEAN
 Com::Objsys::Asn1::Runtime::Asn1Type, 220
 BooleanHolder
 Com::Objsys::Asn1::Runtime::BooleanHolder, 277
 ByteArrayInputStream
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 80
 ByteCount
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 72

 CanRead
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 171
 CanSeek
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 171
 CanWrite
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 171
 Capture
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 68
 Century
 Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, 91
 Characters
 Com::Objsys::Asn1::Runtime::Asn1NamedEventHandler, 121
 Com::Objsys::Asn1::Runtime::Asn1OpenType::SaxHandler, 283
 Com::Objsys::Asn1::Runtime::Asn1TraceHandler, 205
 ChoiceID
 Com::Objsys::Asn1::Runtime::Asn1Choice, 61
 choiceID
 Com::Objsys::Asn1::Runtime::Asn1Choice, 61
 choiceIndex
 Com::Objsys::Asn1::Runtime::Asn1ChoiceExt, 65
 ClassMask
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200
 Clear
 Com::Objsys::Asn1::Runtime::Asn1BitString, 25
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 244
 Close
 Com::Objsys::Asn1::Runtime::Asn1InputStream, 99
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 168
 Com::Objsys::Asn1::Runtime, 3
 Com::Objsys::Asn1::Runtime::Asn18BitCharString, 7
 Asn18BitCharString, 8
 BITSPERCHAR_A, 11
 BITSPERCHAR_U, 11
 Decode, 8, 9
 Encode, 9–11
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 13
 _TAG, 21
 Asn1BigInteger, 14
 Decode, 14, 15
 DecodeSigned, 15
 DecodeUnsigned, 16
 DecodeValue, 16
 DecodeXER, 16
 DecodeXML, 16
 Encode, 16–18
 EncodeAttribute, 19
 EncodeSigned, 19
 EncodeUnsigned, 19
 EncodeValue, 20
 Equals, 20
 GetHashCode, 20
 mValue, 21
 ToString, 20
 Com::Objsys::Asn1::Runtime::Asn1BitString, 22
 _TAG, 32
 Asn1BitString, 23, 24
 Clear, 25
 Decode, 25, 26
 DecodeContent, 26
 DecodeXER, 26
 DecodeXML, 26
 Encode, 27–29
 EncodeContent, 30
 Equals, 30

- Get, 30
- GetHashCode, 31
- GetOerEffectiveMin, 31
- IsNamedBitStr, 31
- Length, 33
- mStringFormat, 32
- mValue, 33
- numbits, 33
- Set, 31
- StringFormat, 23
- this, 33
- ToBoolArray, 32
- ToHexString, 32
- toInputStream, 32
- ToString, 32
- trimZeroBits, 33
- Com::Objsys::Asn1::Runtime::Asn1BMPString, 34
 - _TAG, 39
 - Asn1BMPString, 34
 - BITSPERCHAR, 39
 - Decode, 35, 36
 - Encode, 36–39
- Com::Objsys::Asn1::Runtime::Asn1Boolean, 40
 - _TAG, 46
 - Asn1Boolean, 41
 - Decode, 41
 - DecodeXER, 42
 - DecodeXML, 42
 - Encode, 42–44
 - EncodeAttribute, 44
 - Equals, 45
 - FALSE_VALUE, 46
 - GetHashCode, 45
 - mValue, 46
 - setTrueEncodedByte, 45
 - ToString, 45
 - TRUE_VALUE, 46
- Com::Objsys::Asn1::Runtime::Asn1CharRange, 47
 - Asn1CharRange, 47
 - GetCharAtIndex, 48
 - GetCharIndex, 48
 - MaxValue, 49
 - mLower, 49
 - mUpper, 49
 - validate, 48
- Com::Objsys::Asn1::Runtime::Asn1CharSet, 50
 - Asn1CharSet, 50
 - GetCharAtIndex, 50
 - GetCharIndex, 51
 - GetNumBitsPerChar, 51
 - mABitsPerChar, 52
 - MaxValue, 52
 - mUBitsPerChar, 52
 - validate, 51
- Com::Objsys::Asn1::Runtime::Asn1CharString, 53
 - Asn1CharString, 54
 - Decode, 54, 55
 - DecodeByteToChar, 55
 - DecodeXER, 55
 - DecodeXML, 55
 - Encode, 55–57
 - Equals, 57, 58
 - GetHashCode, 58
 - Length, 59
 - mStringBuffer, 59
 - mValue, 59
 - ToString, 58
 - validate, 58
- Com::Objsys::Asn1::Runtime::Asn1Choice, 60
 - Asn1Choice, 60
 - ChoiceID, 61
 - choiceID, 61
 - element, 61
 - ElemName, 61
 - Equals, 60
 - GetElement, 60
 - GetHashCode, 61
 - SetElement, 61
- Com::Objsys::Asn1::Runtime::Asn1ChoiceExt, 63
 - choiceIndex, 65
 - Decode, 63
 - Encode, 63, 64
 - tag, 65
- Com::Objsys::Asn1::Runtime::Asn1ConsVioException, 66
 - Asn1ConsVioException, 66
- Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 67
 - AddCaptureBuffer, 68
 - ByteCount, 72
 - Capture, 68
 - DecodeIntValue, 68
 - DecodeOIDContents, 68
 - DecodeRelOIDContents, 68
 - GetInputStream, 69
 - HexDump, 69
 - Init, 69
 - LazyOpenTypeDecode, 72
 - Mark, 69
 - mByteCount, 71
 - Read, 69, 70
 - Read2Bytes, 70
 - Read4Bytes, 70
 - ReadByte, 70
 - RemoveCaptureBuffer, 71
 - Reset, 71
 - SetInputStream, 71
 - Skip, 71
- Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, 73

- Asn1DiscreteCharSet, 73
- GetCharAtIndex, 73
- GetCharIndex, 74
- helpValidate, 74
- Max Value, 75
- validate, 74
- Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 76
 - BinDump, 77
 - ByteArrayInputStream, 80
 - Copy, 77
 - EncodeIntSigned, 77
 - EncodeIntUnsigned, 78
 - GetInputStream, 78
 - GetMinimalOctetsSigned, 78
 - GetMinimalOctetsUnsigned, 78
 - GetOutputStream, 79
 - HexDump, 79
 - MsgCopy, 80
 - MsgLength, 80
 - mSizeIncrement, 79
 - Reset, 79
 - SIZE_INCREMENT, 79
 - Write, 79
- Com::Objsys::Asn1::Runtime::Asn1EndOfBufferException, 81
 - Asn1EndOfBufferException, 81
- Com::Objsys::Asn1::Runtime::Asn1Enumerated, 82
 - _TAG, 86
 - Asn1Enumerated, 83
 - Encode, 83–85
 - Equals, 85
 - GetHashCode, 85
 - mValue, 86
 - ParseValue, 86
 - ToString, 86
 - UNDEFINED, 86
 - Value, 87
- Com::Objsys::Asn1::Runtime::Asn1Exception, 88
 - Asn1Exception, 88
- Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, 89
 - _TAG, 91
 - Asn1GeneralizedTime, 89, 90
 - Century, 91
 - CompareTo, 90
 - CompileString, 90
 - Decode, 90
 - Encode, 90, 91
 - ParseString, 91
- Com::Objsys::Asn1::Runtime::Asn1GeneralString, 93
 - _TAG, 94
 - Asn1GeneralString, 93
 - Decode, 93
 - Encode, 93, 94
- Com::Objsys::Asn1::Runtime::Asn1GraphicString, 95
 - _TAG, 96
 - Asn1GraphicString, 95
 - Decode, 95
 - Encode, 95, 96
- Com::Objsys::Asn1::Runtime::Asn1IA5String, 97
 - _TAG, 98
 - Asn1IA5String, 97
 - Decode, 97
 - Encode, 97, 98
- Com::Objsys::Asn1::Runtime::Asn1InputStream, 99
 - Available, 99
 - Close, 99
 - Mark, 99
 - MarkSupported, 99
 - Reset, 99
 - Skip, 100
- Com::Objsys::Asn1::Runtime::Asn1Integer, 101
 - _TAG, 112
 - Asn1Integer, 102
 - Decode, 102–104
 - Decode16Bit, 104
 - Decode32Bit, 104
 - Decode8Bit, 104
 - DecodeSigned, 104
 - DecodeUnsigned, 104, 105
 - DecodeValue, 105
 - DecodeXER, 105
 - DecodeXML, 105
 - Encode, 105–109
 - Encode16Bit, 109
 - Encode32Bit, 109
 - Encode8Bit, 109
 - EncodeAttribute, 109
 - EncodeSigned, 110
 - EncodeUnsigned, 110
 - EncodeValue, 110
 - Equals, 110, 111
 - GetBitCount, 111
 - GetHashCode, 111
 - GetUnsignedBitCount, 111, 112
 - mValue, 112
 - ToString, 112
- Com::Objsys::Asn1::Runtime::Asn1InvalidArgException, 113
 - Asn1InvalidArgException, 113
- Com::Objsys::Asn1::Runtime::Asn1InvalidChoiceOptionException, 114
 - Asn1InvalidChoiceOptionException, 114
- Com::Objsys::Asn1::Runtime::Asn1InvalidEnumException, 115
 - Asn1InvalidEnumException, 115
- Com::Objsys::Asn1::Runtime::Asn1InvalidLengthException, 116
 - Asn1InvalidLengthException, 116

Com::Objsys::Asn1::Runtime::Asn1InvalidObjectIDException, [_TAG, 149](#)
 [117](#)
 Asn1InvalidObjectIDException, [117](#)
 Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, [118](#)
 AddNamedEventHandler, [118](#)
 EventHandlerList, [119](#)
 GetInputStream, [118](#)
 InvokeCharacters, [118](#)
 InvokeEndElement, [118](#)
 InvokeStartElement, [119](#)
 Com::Objsys::Asn1::Runtime::Asn1MissingRequiredException, [120](#)
 Asn1MissingRequiredException, [120](#)
 Com::Objsys::Asn1::Runtime::Asn1NamedEventHandler, [121](#)
 Characters, [121](#)
 EndElement, [121](#)
 StartElement, [121](#)
 Com::Objsys::Asn1::Runtime::Asn1Null, [123](#)
 _TAG, [126](#)
 Decode, [123, 124](#)
 DecodeXER, [124](#)
 DecodeXML, [124](#)
 Encode, [124–126](#)
 Equals, [126](#)
 NULL_VALUE, [126](#)
 ToString, [126](#)
 Com::Objsys::Asn1::Runtime::Asn1NumericString, [128](#)
 _TAG, [130](#)
 Asn1NumericString, [128](#)
 Decode, [128, 129](#)
 Encode, [129, 130](#)
 Com::Objsys::Asn1::Runtime::Asn1ObjectDescriptor, [131](#)
 _TAG, [132](#)
 Asn1ObjectDescriptor, [131](#)
 Decode, [131](#)
 Encode, [131, 132](#)
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, [133](#)
 _TAG, [138](#)
 Append, [134](#)
 Asn1ObjectIdentifier, [134](#)
 Decode, [134](#)
 DecodeXER, [135](#)
 DecodeXML, [135](#)
 Encode, [135–137](#)
 Equals, [137](#)
 GetHashCode, [137](#)
 MAXSUBIDS, [138](#)
 mValue, [138](#)
 ToString, [138](#)
 ToXMLValue, [138](#)
 Validate, [138](#)
 Com::Objsys::Asn1::Runtime::Asn1OctetString, [140](#)
 Asn1OctetString, [141](#)
 CompareTo, [142](#)
 Decode, [142, 143](#)
 DecodeContent, [143](#)
 DecodeXER, [143](#)
 DecodeXML, [144](#)
 Encode, [144–147](#)
 EncodeAttribute, [147](#)
 EncodeBase64Binary, [147](#)
 EncodeContent, [148](#)
 Equals, [148](#)
 GetHashCode, [148](#)
 GetMderLength, [149](#)
 Length, [149](#)
 mValue, [149](#)
 toInputStream, [149](#)
 ToString, [149](#)
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, [151](#)
 Asn1TypeName, [156](#)
 Decode, [151, 152](#)
 DecodeComponent, [152](#)
 DecodeEventComponent, [152](#)
 DecodeExtension, [152](#)
 DecodeOpenType, [152, 153](#)
 Encode, [153–155](#)
 EncodeExtBits, [155](#)
 HasPresentExtensions, [155](#)
 mValue, [156](#)
 SetOpenType, [155](#)
 ShrinkArray, [155](#)
 ToString, [155](#)
 Com::Objsys::Asn1::Runtime::Asn1OpenType, [157](#)
 Asn1OpenType, [158, 159](#)
 Asn1TypeName, [166](#)
 dataEncoding, [166](#)
 Decode, [160](#)
 DecodeExtension, [161](#)
 Encode, [161–163](#)
 EncodeAsExtension, [163, 164](#)
 GetCharData, [164](#)
 GetDataEncoding, [164](#)
 GetSaxHandler, [164](#)
 mEncodeBuffer, [166](#)
 mLength, [166](#)
 SetBinaryData, [165](#)
 SetCharData, [165](#)
 ToString, [165](#)
 Com::Objsys::Asn1::Runtime::Asn1OpenType::SaxHandler, [283](#)
 Characters, [283](#)
 EndElement, [283](#)
 StartElement, [283](#)
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [167](#)

- Asn1OutputStream, 167
- CanRead, 171
- CanSeek, 171
- CanWrite, 171
- Close, 168
- Context, 171
- Flush, 168
- Length, 171
- os, 171
- Position, 172
- Read, 168
- Seek, 168
- SetLength, 169
- Write, 169
- Write2Bytes, 170
- Write4Bytes, 170
- WriteByte, 170
- Com::Objsys::Asn1::Runtime::Asn1PrintableString, 173
 - _TAG, 174
 - Asn1PrintableString, 173
 - Decode, 173
 - Encode, 173, 174
- Com::Objsys::Asn1::Runtime::Asn1Real, 175
 - _TAG, 182
 - Asn1Real, 176
 - Decode, 176, 177
 - DecodeDouble, 177
 - DecodeSingle, 177
 - DecodeXER, 177
 - DecodeXML, 178
 - Encode, 178–180
 - EncodeAttribute, 180
 - EncodeDouble, 180
 - EncodeSingle, 181
 - EncodeValue, 181
 - Equals, 181
 - GetHashCode, 181
 - mValue, 182
 - NormalizedRealValueToString, 181
 - ToString, 182
- Com::Objsys::Asn1::Runtime::Asn1Real10, 183
 - _TAG, 188
 - Asn1Real10, 183
 - ConvertToDecimal, 184
 - ConvertToNR3Form, 184
 - Decode, 184
 - Encode, 184–187
 - EncodeAttribute, 187
 - GetNumberForm, 187
- Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 189
 - _TAG, 193
 - Asn1RelativeOID, 189
 - Decode, 190
 - DecodeXER, 190
 - DecodeXML, 190
 - Encode, 190–192
 - Validate, 192
- Com::Objsys::Asn1::Runtime::Asn1SeqOrderException, 194
 - Asn1SeqOrderException, 194
- Com::Objsys::Asn1::Runtime::Asn1Status, 195
 - INDEFLEN, 195
- Com::Objsys::Asn1::Runtime::Asn1T61String, 196
 - _TAG, 197
 - Asn1T61String, 196
 - Decode, 196
 - Encode, 196, 197
- Com::Objsys::Asn1::Runtime::Asn1Tag, 198
 - APPL, 200
 - Asn1Tag, 199
 - Bit8Mask, 200
 - ClassMask, 200
 - CONS, 200
 - Constructed, 202
 - CTXT, 200
 - ENUM, 200
 - EOC, 200
 - Equals, 199
 - EXPL, 200
 - EXTIDCODE, 200
 - FormMask, 200
 - IDMask, 201
 - IMPL, 201
 - IsEOC, 199
 - L7BitsMask, 201
 - mClass, 201
 - mForm, 201
 - mIDCode, 201
 - PRIM, 201
 - PRIV, 201
 - SEQUENCE, 201
 - SET, 201
 - ToString, 199
 - UNIV, 201
- Com::Objsys::Asn1::Runtime::Asn1Time, 203
 - Asn1Time, 203
 - Decode, 203
 - Encode, 203, 204
 - January, 204
- Com::Objsys::Asn1::Runtime::Asn1TraceHandler, 205
 - Asn1TraceHandler, 205
 - Characters, 205
 - EndElement, 205
 - StartElement, 206
- Com::Objsys::Asn1::Runtime::Asn1Type, 207
 - _SetKey, 209
 - _SetKey2, 209
 - _TAG, 220

- AsnTypeName, 223
- BIT_STRING, 220
- BMPString, 220
- BOOLEAN, 220
- DATE, 220
- Decode, 209–211
- DecodeXML, 212
- Encode, 212–216
- EncodeAsOpenType, 216
- EncodeAttribute, 217
- ENUMERATED, 220
- EOC, 220
- Equals, 217
- EXTERNAL, 220
- GeneralString, 220
- GeneralTime, 220
- GetNonParameterizedTypeName, 217
- GetTypeName, 217
- GraphicString, 221
- IA5String, 221
- Indent, 217
- INTEGER, 221
- IsOpenType, 217
- Length, 223
- MatchTag, 218
- MatchTypeName, 218
- NULL, 221
- NumericString, 221
- OBJECT_IDENTIFIER, 221
- ObjectDescriptor, 221
- OCTET_STRING, 221
- OpenType, 221
- Pdiag, 219
- Print, 219
- PrintableString, 221
- REAL, 221
- RELATIVE_OID_IRI, 221
- RelativeOID, 222
- SEQUENCE, 222
- SET, 222
- SetNonParameterizedTypeName, 219
- SetOpenType, 219
- T61String, 222
- TeletexString, 222
- TIME, 222
- UniversalString, 222
- UTCTime, 222
- UTF8String, 222
- VideotexString, 222
- VisibleString, 222
- Com::Objsys::Asn1::Runtime::Asn1TypeIF, 224
 - Decode, 224, 225
 - Encode, 225–228
 - IsOpenType, 228
 - Print, 228
 - SetOpenType, 229
- Com::Objsys::Asn1::Runtime::Asn1UniversalString, 230
 - _TAG, 241
 - Asn1UniversalString, 231
 - BITSPERCHAR, 241
 - Decode, 232–234
 - DecodeXER, 234
 - DecodeXML, 234
 - Encode, 234–240
 - EncodeData, 240
 - Equals, 240
 - GetHashCode, 241
 - Length, 242
 - mStringBuffer, 241
 - mValue, 241
 - ToString, 241
 - validate, 241
- Com::Objsys::Asn1::Runtime::Asn1UTCTime, 243
 - _TAG, 246
 - Asn1UTCTime, 243, 244
 - Clear, 244
 - CompareTo, 244
 - CompileString, 244
 - Decode, 244
 - Encode, 245
 - Fraction, 246
 - Init, 245
 - ParseString, 245
 - SetTime, 246
 - Year, 246
- Com::Objsys::Asn1::Runtime::Asn1UTF8String, 247
 - _TAG, 253
 - Asn1UTF8String, 247
 - Decode, 248, 249
 - DecodeUTF8, 249
 - Encode, 249–252
 - SetAnyAttribute, 253
- Com::Objsys::Asn1::Runtime::Asn1Util, 254
 - BCDToString, 254
 - DecodeBase64Array, 254
 - EncodeBase64Array, 255
 - GetAddressBytes, 255
 - GetBytesCount, 255
 - GetUlongBytesCount, 255
 - IsLimited, 256
 - StringToBCD, 256
 - StringToTBCD, 256
 - StripWhitespace, 256
 - TBCDToString, 257
 - ToArray, 257
 - ToByteArray, 257
 - ToCharArray, 257
 - ToHexString, 258

- URShift, [258](#), [259](#)
- WriteStackTrace, [259](#)
- Com::Objsys::Asn1::Runtime::Asn1Value, [261](#)
 - ParseString, [261](#)
 - StringEqualsBytes, [261](#), [262](#)
- Com::Objsys::Asn1::Runtime::Asn1ValueParseException, [263](#)
 - Asn1ValueParseException, [263](#)
- Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString, [264](#)
 - Asn1VarWidthCharString, [264](#)
 - BITSPERCHAR_A, [267](#)
 - BITSPERCHAR_U, [267](#)
 - Decode, [265](#)
 - Encode, [265](#), [266](#)
- Com::Objsys::Asn1::Runtime::Asn1VideotexString, [268](#)
 - _TAG, [269](#)
 - Asn1VideotexString, [268](#)
 - Decode, [268](#)
 - Encode, [269](#)
- Com::Objsys::Asn1::Runtime::Asn1VisibleString, [270](#)
 - _TAG, [271](#)
 - Asn1VisibleString, [270](#)
 - Decode, [270](#)
 - Encode, [270](#), [271](#)
- Com::Objsys::Asn1::Runtime::BigInteger, [272](#)
 - Add, [273](#)
 - BigInteger, [272](#), [273](#)
 - BitLength, [273](#)
 - CompareTo, [273](#)
 - Equals, [274](#)
 - GetData, [274](#)
 - GetHashCode, [274](#)
 - Init, [274](#)
 - IsNegative, [275](#)
 - LongValue, [275](#)
 - operator BigInteger, [275](#)
 - SecureDelete, [275](#)
 - SetData, [275](#)
 - Subtract, [276](#)
 - ToString, [276](#)
- Com::Objsys::Asn1::Runtime::BooleanHolder, [277](#)
 - BooleanHolder, [277](#)
 - mValue, [277](#)
- Com::Objsys::Asn1::Runtime::Diag, [278](#)
 - HexDump, [278](#), [279](#)
 - Instance, [279](#)
 - IsEnabled, [279](#)
 - Println, [279](#)
 - PrintStream, [281](#)
 - Prtln, [280](#)
 - SetEnabled, [280](#)
 - SetTraceLevel, [281](#)
 - SetTraceLevel2, [281](#)
- Com::Objsys::Asn1::Runtime::IntHolder, [282](#)
 - IntHolder, [282](#)
 - mValue, [282](#)
- Com::Objsys::Asn1::Runtime::StringBufferExt, [284](#)
 - Replace, [284](#)
- Com::Objsys::Asn1::Runtime::Tokenizer, [285](#)
 - Count, [287](#)
 - Current, [287](#)
 - HasMoreTokens, [286](#)
 - MoveNext, [286](#)
 - NextToken, [286](#)
 - RemainingString, [286](#)
 - Reset, [286](#)
 - Tokenizer, [285](#)
- CompareTo
 - Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, [90](#)
 - Com::Objsys::Asn1::Runtime::Asn1OctetString, [142](#)
 - Com::Objsys::Asn1::Runtime::Asn1UTCTime, [244](#)
 - Com::Objsys::Asn1::Runtime::BigInteger, [273](#)
- CompileString
 - Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, [90](#)
 - Com::Objsys::Asn1::Runtime::Asn1UTCTime, [244](#)
- CONS
 - Com::Objsys::Asn1::Runtime::Asn1Tag, [200](#)
- Constructed
 - Com::Objsys::Asn1::Runtime::Asn1Tag, [202](#)
- Context
 - Com::Objsys::Asn1::Runtime::Asn1OutputStream, [171](#)
- ConvertToDecimal
 - Com::Objsys::Asn1::Runtime::Asn1Real10, [184](#)
- ConvertToNR3Form
 - Com::Objsys::Asn1::Runtime::Asn1Real10, [184](#)
- Copy
 - Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, [77](#)
- Count
 - Com::Objsys::Asn1::Runtime::Tokenizer, [287](#)
- CTXT
 - Com::Objsys::Asn1::Runtime::Asn1Tag, [200](#)
- Current
 - Com::Objsys::Asn1::Runtime::Tokenizer, [287](#)
- dataEncoding
 - Com::Objsys::Asn1::Runtime::Asn1OpenType, [166](#)
- DATE
 - Com::Objsys::Asn1::Runtime::Asn1Type, [220](#)
- Decode
 - Com::Objsys::Asn1::Runtime::Asn18BitCharString, [8](#), [9](#)

Com::Objsys::Asn1::Runtime::Asn1BigInteger, 14, 15
 Com::Objsys::Asn1::Runtime::Asn1BitString, 25, 26
 Com::Objsys::Asn1::Runtime::Asn1BMPString, 35, 36
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 41
 Com::Objsys::Asn1::Runtime::Asn1CharString, 54, 55
 Com::Objsys::Asn1::Runtime::Asn1ChoiceExt, 63
 Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, 90
 Com::Objsys::Asn1::Runtime::Asn1GeneralString, 93
 Com::Objsys::Asn1::Runtime::Asn1GraphicString, 95
 Com::Objsys::Asn1::Runtime::Asn1IA5String, 97
 Com::Objsys::Asn1::Runtime::Asn1Integer, 102–104
 Com::Objsys::Asn1::Runtime::Asn1Null, 123, 124
 Com::Objsys::Asn1::Runtime::Asn1NumericString, 128, 129
 Com::Objsys::Asn1::Runtime::Asn1ObjectDescriptor, 131
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 134
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 142, 143
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 151, 152
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 160
 Com::Objsys::Asn1::Runtime::Asn1PrintableString, 173
 Com::Objsys::Asn1::Runtime::Asn1Real, 176, 177
 Com::Objsys::Asn1::Runtime::Asn1Real10, 184
 Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 190
 Com::Objsys::Asn1::Runtime::Asn1T61String, 196
 Com::Objsys::Asn1::Runtime::Asn1Time, 203
 Com::Objsys::Asn1::Runtime::Asn1Type, 209–211
 Com::Objsys::Asn1::Runtime::Asn1TypeIF, 224, 225
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 232–234
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 244
 Com::Objsys::Asn1::Runtime::Asn1UTF8String, 248, 249
 Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString, 265
 Com::Objsys::Asn1::Runtime::Asn1VideotexString, 268
 Com::Objsys::Asn1::Runtime::Asn1VisibleString, 270
 Decode16Bit
 Com::Objsys::Asn1::Runtime::Asn1Integer, 104
 Decode32Bit
 Com::Objsys::Asn1::Runtime::Asn1Integer, 104
 Decode8Bit
 Com::Objsys::Asn1::Runtime::Asn1Integer, 104
 DecodeBase64Array
 Com::Objsys::Asn1::Runtime::Asn1Util, 254
 DecodeByteToChar
 Com::Objsys::Asn1::Runtime::Asn1CharString, 55
 DecodeComponent
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 152
 DecodeContent
 Com::Objsys::Asn1::Runtime::Asn1BitString, 26
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 143
 DecodeDouble
 Com::Objsys::Asn1::Runtime::Asn1Real, 177
 DecodeEventComponent
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 152
 DecodeExtension
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 152
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 161
 DecodeIntValue
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 68
 DecodeOIDContents
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 68
 DecodeOpenType
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 152, 153
 DecodeRelOIDContents
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 68
 DecodeSigned
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 15
 Com::Objsys::Asn1::Runtime::Asn1Integer, 104
 DecodeSingle
 Com::Objsys::Asn1::Runtime::Asn1Real, 177
 DecodeUnsigned
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 16
 Com::Objsys::Asn1::Runtime::Asn1Integer, 104, 105
 DecodeUTF8
 Com::Objsys::Asn1::Runtime::Asn1UTF8String, 249
 DecodeValue
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 16
 Com::Objsys::Asn1::Runtime::Asn1Integer, 105
 DecodeXER
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 16
 Com::Objsys::Asn1::Runtime::Asn1BitString, 26
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 42
 Com::Objsys::Asn1::Runtime::Asn1CharString, 55

Com::Objsys::Asn1::Runtime::Asn1Integer, 105
 Com::Objsys::Asn1::Runtime::Asn1Null, 124
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 135
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 143
 Com::Objsys::Asn1::Runtime::Asn1Real, 177
 Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 190
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 234
 DecodeXML
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 16
 Com::Objsys::Asn1::Runtime::Asn1BitString, 26
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 42
 Com::Objsys::Asn1::Runtime::Asn1CharString, 55
 Com::Objsys::Asn1::Runtime::Asn1Integer, 105
 Com::Objsys::Asn1::Runtime::Asn1Null, 124
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 135
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 144
 Com::Objsys::Asn1::Runtime::Asn1Real, 178
 Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 190
 Com::Objsys::Asn1::Runtime::Asn1Type, 212
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 234
 element
 Com::Objsys::Asn1::Runtime::Asn1Choice, 61
 ElemName
 Com::Objsys::Asn1::Runtime::Asn1Choice, 61
 Encode
 Com::Objsys::Asn1::Runtime::Asn18BitCharString, 9–11
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 16–18
 Com::Objsys::Asn1::Runtime::Asn1BitString, 27–29
 Com::Objsys::Asn1::Runtime::Asn1BMPString, 36–39
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 42–44
 Com::Objsys::Asn1::Runtime::Asn1CharString, 55–57
 Com::Objsys::Asn1::Runtime::Asn1ChoiceExt, 63, 64
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, 83–85
 Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, 90, 91
 Com::Objsys::Asn1::Runtime::Asn1GeneralString, 93, 94
 Com::Objsys::Asn1::Runtime::Asn1GraphicString, 95, 96
 Com::Objsys::Asn1::Runtime::Asn1IA5String, 97, 98
 Com::Objsys::Asn1::Runtime::Asn1Integer, 105–109
 Com::Objsys::Asn1::Runtime::Asn1Null, 124–126
 Com::Objsys::Asn1::Runtime::Asn1NumericString, 129, 130
 Com::Objsys::Asn1::Runtime::Asn1ObjectDescriptor, 131, 132
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 135–137
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 144–147
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 153–155
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 161–163
 Com::Objsys::Asn1::Runtime::Asn1PrintableString, 173, 174
 Com::Objsys::Asn1::Runtime::Asn1Real, 178–180
 Com::Objsys::Asn1::Runtime::Asn1Real10, 184–187
 Com::Objsys::Asn1::Runtime::Asn1RelativeOID, 190–192
 Com::Objsys::Asn1::Runtime::Asn1T61String, 196, 197
 Com::Objsys::Asn1::Runtime::Asn1Time, 203, 204
 Com::Objsys::Asn1::Runtime::Asn1Type, 212–216
 Com::Objsys::Asn1::Runtime::Asn1TypeIF, 225–228
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 234–240
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 245
 Com::Objsys::Asn1::Runtime::Asn1UTF8String, 249–252
 Com::Objsys::Asn1::Runtime::Asn1VarWidthCharString, 265, 266
 Com::Objsys::Asn1::Runtime::Asn1VideotexString, 269
 Com::Objsys::Asn1::Runtime::Asn1VisibleString, 270, 271
 Encode16Bit
 Com::Objsys::Asn1::Runtime::Asn1Integer, 109
 Encode32Bit
 Com::Objsys::Asn1::Runtime::Asn1Integer, 109
 Encode8Bit
 Com::Objsys::Asn1::Runtime::Asn1Integer, 109
 EncodeAsExtension
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 163, 164
 EncodeAsOpenType
 Com::Objsys::Asn1::Runtime::Asn1Type, 216

EncodeAttribute
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 19
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 44
 Com::Objsys::Asn1::Runtime::Asn1Integer, 109
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 147
 Com::Objsys::Asn1::Runtime::Asn1Real, 180
 Com::Objsys::Asn1::Runtime::Asn1Real10, 187
 Com::Objsys::Asn1::Runtime::Asn1Type, 217

EncodeBase64Array
 Com::Objsys::Asn1::Runtime::Asn1Util, 255

EncodeBase64Binary
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 147

EncodeContent
 Com::Objsys::Asn1::Runtime::Asn1BitString, 30
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 148

EncodeData
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 240

EncodeDouble
 Com::Objsys::Asn1::Runtime::Asn1Real, 180

EncodeExtBits
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 155

EncodeIntSigned
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 77

EncodeIntUnsigned
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 78

EncodeSigned
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 19
 Com::Objsys::Asn1::Runtime::Asn1Integer, 110

EncodeSingle
 Com::Objsys::Asn1::Runtime::Asn1Real, 181

EncodeUnsigned
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 19
 Com::Objsys::Asn1::Runtime::Asn1Integer, 110

EncodeValue
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 20
 Com::Objsys::Asn1::Runtime::Asn1Integer, 110
 Com::Objsys::Asn1::Runtime::Asn1Real, 181

EndElement
 Com::Objsys::Asn1::Runtime::Asn1NamedEventHandler, 121
 Com::Objsys::Asn1::Runtime::Asn1OpenType::SaxHandler, 283
 Com::Objsys::Asn1::Runtime::Asn1TraceHandler, 205

ENUM
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200

ENUMERATED
 Com::Objsys::Asn1::Runtime::Asn1Type, 220

EOC
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200
 Com::Objsys::Asn1::Runtime::Asn1Type, 220

Equals
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 20
 Com::Objsys::Asn1::Runtime::Asn1BitString, 30
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 45
 Com::Objsys::Asn1::Runtime::Asn1CharString, 57, 58
 Com::Objsys::Asn1::Runtime::Asn1Choice, 60
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, 85
 Com::Objsys::Asn1::Runtime::Asn1Integer, 110, 111
 Com::Objsys::Asn1::Runtime::Asn1Null, 126
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 137
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 148
 Com::Objsys::Asn1::Runtime::Asn1Real, 181
 Com::Objsys::Asn1::Runtime::Asn1Tag, 199
 Com::Objsys::Asn1::Runtime::Asn1Type, 217
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 240
 Com::Objsys::Asn1::Runtime::BigInteger, 274

EventHandlerList
 Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, 119

EXPL
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200

EXTERNAL
 Com::Objsys::Asn1::Runtime::Asn1Type, 220

EXTIDCODE
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200

FALSE_VALUE
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 46

Flush
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 168

FormMask
 Com::Objsys::Asn1::Runtime::Asn1Tag, 200

Fraction
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 246

GeneralString
 Com::Objsys::Asn1::Runtime::Asn1Type, 220

GeneralTime
 Com::Objsys::Asn1::Runtime::Asn1Type, 220

Get
 Com::Objsys::Asn1::Runtime::Asn1BitString, 30

GetAddressBytes
 Com::Objsys::Asn1::Runtime::Asn1Util, 255

GetBitCount
 Com::Objsys::Asn1::Runtime::Asn1Integer, 111

GetBytesCount
 Com::Objsys::Asn1::Runtime::Asn1Util, 255

GetCharAtIndex
 Com::Objsys::Asn1::Runtime::Asn1CharRange, 48
 Com::Objsys::Asn1::Runtime::Asn1CharSet, 50
 Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, 73

GetCharData
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 164

GetCharIndex
 Com::Objsys::Asn1::Runtime::Asn1CharRange, 48
 Com::Objsys::Asn1::Runtime::Asn1CharSet, 51
 Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, 74

GetData
 Com::Objsys::Asn1::Runtime::BigInteger, 274

GetDataEncoding
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 164

GetElement
 Com::Objsys::Asn1::Runtime::Asn1Choice, 60

GetHashCode
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 20
 Com::Objsys::Asn1::Runtime::Asn1BitString, 31
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 45
 Com::Objsys::Asn1::Runtime::Asn1CharString, 58
 Com::Objsys::Asn1::Runtime::Asn1Choice, 61
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, 85
 Com::Objsys::Asn1::Runtime::Asn1Integer, 111
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 137
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 148
 Com::Objsys::Asn1::Runtime::Asn1Real, 181
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 241
 Com::Objsys::Asn1::Runtime::BigInteger, 274

GetInputStream
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 69
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 78
 Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, 118

GetMderLength
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 149

GetMinimalOctetsSigned
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 78

GetMinimalOctetsUnsigned
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 78

GetNonParameterizedTypeName
 Com::Objsys::Asn1::Runtime::Asn1Type, 217

GetNumberForm
 Com::Objsys::Asn1::Runtime::Asn1Real10, 187

GetNumBitsPerChar
 Com::Objsys::Asn1::Runtime::Asn1CharSet, 51

GetOerEffectiveMin
 Com::Objsys::Asn1::Runtime::Asn1BitString, 31

GetOutputStream
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 79

GetSaxHandler
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 164

GetTypeName
 Com::Objsys::Asn1::Runtime::Asn1Type, 217

GetUlongBytesCount
 Com::Objsys::Asn1::Runtime::Asn1Util, 255

GetUnsignedBitCount
 Com::Objsys::Asn1::Runtime::Asn1Integer, 111, 112

GraphicString
 Com::Objsys::Asn1::Runtime::Asn1Type, 221

HasMoreTokens
 Com::Objsys::Asn1::Runtime::Tokenizer, 286

HasPresentExtensions
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 155

helpValidate
 Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, 74

HexDump
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 69
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 79
 Com::Objsys::Asn1::Runtime::Diag, 278, 279

IA5String
 Com::Objsys::Asn1::Runtime::Asn1Type, 221

IDMask
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201

IMPL
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201

INDEFLEN
 Com::Objsys::Asn1::Runtime::Asn1Status, 195

Indent
 Com::Objsys::Asn1::Runtime::Asn1Type, 217

Init
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 69
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 245
 Com::Objsys::Asn1::Runtime::BigInteger, 274

Instance
 Com::Objsys::Asn1::Runtime::Diag, 279

INTEGER
 Com::Objsys::Asn1::Runtime::Asn1Type, 221

IntHolder
 Com::Objsys::Asn1::Runtime::IntHolder, 282
 InvokeCharacters
 Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, 118
 InvokeEndElement
 Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, 118
 InvokeStartElement
 Com::Objsys::Asn1::Runtime::Asn1MessageBuffer, 119
 IsEnabled
 Com::Objsys::Asn1::Runtime::Diag, 279
 IsEOC
 Com::Objsys::Asn1::Runtime::Asn1Tag, 199
 IsLimited
 Com::Objsys::Asn1::Runtime::Asn1Util, 256
 IsNamedBitStr
 Com::Objsys::Asn1::Runtime::Asn1BitString, 31
 IsNegative
 Com::Objsys::Asn1::Runtime::BigInteger, 275
 IsOpenType
 Com::Objsys::Asn1::Runtime::Asn1Type, 217
 Com::Objsys::Asn1::Runtime::Asn1TypeIF, 228

 January
 Com::Objsys::Asn1::Runtime::Asn1Time, 204

 L7BitsMask
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201
 LazyOpenTypeDecode
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 72

 Length
 Com::Objsys::Asn1::Runtime::Asn1BitString, 33
 Com::Objsys::Asn1::Runtime::Asn1CharString, 59
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 149
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 171
 Com::Objsys::Asn1::Runtime::Asn1Type, 223
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 242

 LongValue
 Com::Objsys::Asn1::Runtime::BigInteger, 275

 mABitsPerChar
 Com::Objsys::Asn1::Runtime::Asn1CharSet, 52
 Mark
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 69
 Com::Objsys::Asn1::Runtime::Asn1InputStream, 99
 MarkSupported
 Com::Objsys::Asn1::Runtime::Asn1InputStream, 99
 MatchTag
 Com::Objsys::Asn1::Runtime::Asn1Type, 218
 MatchTypeName
 Com::Objsys::Asn1::Runtime::Asn1Type, 218
 MAXSUBIDS
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, 138
 MaxValue
 Com::Objsys::Asn1::Runtime::Asn1CharRange, 49
 Com::Objsys::Asn1::Runtime::Asn1CharSet, 52
 Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, 75
 mByteCount
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 71
 mClass
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201
 mEncodeBuffer
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 166
 mForm
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201
 mIDCode
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201
 mLength
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 166
 mLower
 Com::Objsys::Asn1::Runtime::Asn1CharRange, 49
 MoveNext
 Com::Objsys::Asn1::Runtime::Tokenizer, 286
 MsgCopy
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 80
 MsgLength
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 80
 mSizeIncrement
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 79
 mStringBuffer
 Com::Objsys::Asn1::Runtime::Asn1CharString, 59
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, 241
 mStringFormat
 Com::Objsys::Asn1::Runtime::Asn1BitString, 32
 mUBitsPerChar
 Com::Objsys::Asn1::Runtime::Asn1CharSet, 52
 mUpper
 Com::Objsys::Asn1::Runtime::Asn1CharRange, 49
 mValue
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 21
 Com::Objsys::Asn1::Runtime::Asn1BitString, 33
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 46

Com::Objsys::Asn1::Runtime::Asn1CharString, [59](#)
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, [86](#)
 Com::Objsys::Asn1::Runtime::Asn1Integer, [112](#)
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, [138](#)
 Com::Objsys::Asn1::Runtime::Asn1OctetString, [149](#)
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, [156](#)
 Com::Objsys::Asn1::Runtime::Asn1Real, [182](#)
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, [241](#)
 Com::Objsys::Asn1::Runtime::BooleanHolder, [277](#)
 Com::Objsys::Asn1::Runtime::IntHolder, [282](#)

NextToken
 Com::Objsys::Asn1::Runtime::Tokenizer, [286](#)
 NormalizedRealValueToString
 Com::Objsys::Asn1::Runtime::Asn1Real, [181](#)
 NULL
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 NULL_VALUE
 Com::Objsys::Asn1::Runtime::Asn1Null, [126](#)
 numbits
 Com::Objsys::Asn1::Runtime::Asn1BitString, [33](#)
 NumericString
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)

OBJECT_IDENTIFIER
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 ObjectDescriptor
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 OCTET_STRING
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 OpenType
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 operator BigInteger
 Com::Objsys::Asn1::Runtime::BigInteger, [275](#)

os
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [171](#)

ParseString
 Com::Objsys::Asn1::Runtime::Asn1GeneralizedTime, [91](#)
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, [245](#)
 Com::Objsys::Asn1::Runtime::Asn1Value, [261](#)
 ParseValue
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, [86](#)
 Pdiag
 Com::Objsys::Asn1::Runtime::Asn1Type, [219](#)
 Position
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [172](#)
 PRIM
 Com::Objsys::Asn1::Runtime::Asn1Tag, [201](#)

Print
 Com::Objsys::Asn1::Runtime::Asn1Type, [219](#)
 Com::Objsys::Asn1::Runtime::Asn1TypeIF, [228](#)
 PrintableString
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 Println
 Com::Objsys::Asn1::Runtime::Diag, [279](#)
 PrintStream
 Com::Objsys::Asn1::Runtime::Diag, [281](#)
 PRIV
 Com::Objsys::Asn1::Runtime::Asn1Tag, [201](#)
 Prtln
 Com::Objsys::Asn1::Runtime::Diag, [280](#)

Read
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, [69](#), [70](#)
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [168](#)
 Read2Bytes
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, [70](#)
 Read4Bytes
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, [70](#)
 ReadByte
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, [70](#)

REAL
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 RELATIVE_OID_IRI
 Com::Objsys::Asn1::Runtime::Asn1Type, [221](#)
 RelativeOID
 Com::Objsys::Asn1::Runtime::Asn1Type, [222](#)
 RemainingString
 Com::Objsys::Asn1::Runtime::Tokenizer, [286](#)
 RemoveCaptureBuffer
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, [71](#)
 Replace
 Com::Objsys::Asn1::Runtime::StringBufferExt, [284](#)
 Reset
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, [71](#)
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, [79](#)
 Com::Objsys::Asn1::Runtime::Asn1InputStream, [99](#)
 Com::Objsys::Asn1::Runtime::Tokenizer, [286](#)

SecureDelete
 Com::Objsys::Asn1::Runtime::BigInteger, [275](#)
 Seek
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [168](#)

SEQUENCE
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201
 Com::Objsys::Asn1::Runtime::Asn1Type, 222

SET
 Com::Objsys::Asn1::Runtime::Asn1Tag, 201
 Com::Objsys::Asn1::Runtime::Asn1Type, 222

Set
 Com::Objsys::Asn1::Runtime::Asn1BitString, 31

SetAnyAttribute
 Com::Objsys::Asn1::Runtime::Asn1UTF8String, 253

SetBinaryData
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 165

SetCharData
 Com::Objsys::Asn1::Runtime::Asn1OpenType, 165

SetData
 Com::Objsys::Asn1::Runtime::BigInteger, 275

SetElement
 Com::Objsys::Asn1::Runtime::Asn1Choice, 61

SetEnabled
 Com::Objsys::Asn1::Runtime::Diag, 280

SetInputStream
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 71

SetLength
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, 169

SetNonParameterizedTypeName
 Com::Objsys::Asn1::Runtime::Asn1Type, 219

SetOpenType
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 155
 Com::Objsys::Asn1::Runtime::Asn1Type, 219
 Com::Objsys::Asn1::Runtime::Asn1TypeIF, 229

SetTime
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, 246

SetTraceLevel
 Com::Objsys::Asn1::Runtime::Diag, 281

SetTraceLevel2
 Com::Objsys::Asn1::Runtime::Diag, 281

setTrueEncodedByte
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 45

ShrinkArray
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, 155

SIZE_INCREMENT
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, 79

Skip
 Com::Objsys::Asn1::Runtime::Asn1DecodeBuffer, 71
 Com::Objsys::Asn1::Runtime::Asn1InputStream, 100

StartElement
 Com::Objsys::Asn1::Runtime::Asn1NamedEventHandler, 121

Com::Objsys::Asn1::Runtime::Asn1OpenType::SaxHandler, 283
 Com::Objsys::Asn1::Runtime::Asn1TraceHandler, 206

StringEqualsBytes
 Com::Objsys::Asn1::Runtime::Asn1Value, 261, 262

StringFormat
 Com::Objsys::Asn1::Runtime::Asn1BitString, 23

StringToBCD
 Com::Objsys::Asn1::Runtime::Asn1Util, 256

StringToTBCD
 Com::Objsys::Asn1::Runtime::Asn1Util, 256

StripWhitespace
 Com::Objsys::Asn1::Runtime::Asn1Util, 256

Subtract
 Com::Objsys::Asn1::Runtime::BigInteger, 276

T61String
 Com::Objsys::Asn1::Runtime::Asn1Type, 222

tag
 Com::Objsys::Asn1::Runtime::Asn1ChoiceExt, 65

TBCDToString
 Com::Objsys::Asn1::Runtime::Asn1Util, 257

TeletexString
 Com::Objsys::Asn1::Runtime::Asn1Type, 222

this
 Com::Objsys::Asn1::Runtime::Asn1BitString, 33

TIME
 Com::Objsys::Asn1::Runtime::Asn1Type, 222

ToArray
 Com::Objsys::Asn1::Runtime::Asn1Util, 257

ToBoolArray
 Com::Objsys::Asn1::Runtime::Asn1BitString, 32

ToByteArray
 Com::Objsys::Asn1::Runtime::Asn1Util, 257

ToCharArray
 Com::Objsys::Asn1::Runtime::Asn1Util, 257

ToHexString
 Com::Objsys::Asn1::Runtime::Asn1BitString, 32
 Com::Objsys::Asn1::Runtime::Asn1Util, 258

toInputStream
 Com::Objsys::Asn1::Runtime::Asn1BitString, 32
 Com::Objsys::Asn1::Runtime::Asn1OctetString, 149

Tokenizer
 Com::Objsys::Asn1::Runtime::Tokenizer, 285

ToString
 Com::Objsys::Asn1::Runtime::Asn1BigInteger, 20
 Com::Objsys::Asn1::Runtime::Asn1BitString, 32
 Com::Objsys::Asn1::Runtime::Asn1Boolean, 45
 Com::Objsys::Asn1::Runtime::Asn1CharString, 58
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, 86
 Com::Objsys::Asn1::Runtime::Asn1Integer, 112
 Com::Objsys::Asn1::Runtime::Asn1Null, 126

Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, [138](#)
 Com::Objsys::Asn1::Runtime::Asn1OctetString, [149](#)
 Com::Objsys::Asn1::Runtime::Asn1OpenExt, [155](#)
 Com::Objsys::Asn1::Runtime::Asn1OpenType, [165](#)
 Com::Objsys::Asn1::Runtime::Asn1Real, [182](#)
 Com::Objsys::Asn1::Runtime::Asn1Tag, [199](#)
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, [241](#)
 Com::Objsys::Asn1::Runtime::BigInteger, [276](#)
 ToXMLValue
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, [138](#)
 trimZeroBits
 Com::Objsys::Asn1::Runtime::Asn1BitString, [33](#)
 TRUE_VALUE
 Com::Objsys::Asn1::Runtime::Asn1Boolean, [46](#)

 UNDEFINED
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, [86](#)
 UNIV
 Com::Objsys::Asn1::Runtime::Asn1Tag, [201](#)
 UniversalString
 Com::Objsys::Asn1::Runtime::Asn1Type, [222](#)
 URShift
 Com::Objsys::Asn1::Runtime::Asn1Util, [258](#), [259](#)
 UTCTime
 Com::Objsys::Asn1::Runtime::Asn1Type, [222](#)
 UTF8String
 Com::Objsys::Asn1::Runtime::Asn1Type, [222](#)

 Validate
 Com::Objsys::Asn1::Runtime::Asn1ObjectIdentifier, [138](#)
 Com::Objsys::Asn1::Runtime::Asn1RelativeOID, [192](#)
 validate
 Com::Objsys::Asn1::Runtime::Asn1CharRange, [48](#)
 Com::Objsys::Asn1::Runtime::Asn1CharSet, [51](#)
 Com::Objsys::Asn1::Runtime::Asn1CharString, [58](#)
 Com::Objsys::Asn1::Runtime::Asn1DiscreteCharSet, [74](#)
 Com::Objsys::Asn1::Runtime::Asn1UniversalString, [241](#)

 Value
 Com::Objsys::Asn1::Runtime::Asn1Enumerated, [87](#)
 VideotexString
 Com::Objsys::Asn1::Runtime::Asn1Type, [222](#)
 VisibleString
 Com::Objsys::Asn1::Runtime::Asn1Type, [222](#)

 Write
 Com::Objsys::Asn1::Runtime::Asn1EncodeBuffer, [79](#)
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [169](#)
 Write2Bytes
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [170](#)
 Write4Bytes
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [170](#)
 WriteByte
 Com::Objsys::Asn1::Runtime::Asn1OutputStream, [170](#)
 WriteStackTrace
 Com::Objsys::Asn1::Runtime::Asn1Util, [259](#)
 Year
 Com::Objsys::Asn1::Runtime::Asn1UTCTime, [246](#)