

ASN1C

ASN.1 Compiler
Version 5.8
C# BER/DER/PER/XER
RuntimeReference 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 © 1997-2005 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.

CHANGE HISTORY

Date	Author	Version	Description
11/18/2005	AP	5.8	Initial version

Table of Contents

CLASS DOCUMENTATION	1
COM::OBJSYS::ASN1::RUNTIME::ASN1BERDECODEBUFFER	1
COM::OBJSYS::ASN1::RUNTIME::ASN1BERDECODECONTEXT	5
COM::OBJSYS::ASN1::RUNTIME::ASN1BERENCODEBUFFER	7
COM::OBJSYS::ASN1::RUNTIME::ASN1BERINPUTSTREAM	11
COM::OBJSYS::ASN1::RUNTIME::ASN1BERMESSAGEDUMPHANDLER	13
COM::OBJSYS::ASN1::RUNTIME::ASN1BEROUTPUTSTREAM	14
COM::OBJSYS::ASN1::RUNTIME::ASN1CERINPUTSTREAM	18
COM::OBJSYS::ASN1::RUNTIME::ASN1CEROUTPUTSTREAM	19
COM::OBJSYS::ASN1::RUNTIME::ASN1DERDECODEBUFFER	22
COM::OBJSYS::ASN1::RUNTIME::ASN1DERENCODEBUFFER	22
COM::OBJSYS::ASN1::RUNTIME::ASN1DERINPUTSTREAM	23
COM::OBJSYS::ASN1::RUNTIME::ASN1NOTINSETEXCEPTION	24
COM::OBJSYS::ASN1::RUNTIME::ASN1PERBITFIELD	25
COM::OBJSYS::ASN1::RUNTIME::ASN1PERBITFIELDLIST	26
COM::OBJSYS::ASN1::RUNTIME::ASN1PERBITFIELDPRINTER	28
COM::OBJSYS::ASN1::RUNTIME::ASN1PERDECODEBUFFER	30
COM::OBJSYS::ASN1::RUNTIME::ASN1PERDECODETRACEHANDLER	36
COM::OBJSYS::ASN1::RUNTIME::ASN1PERENCODEBUFFER	37
COM::OBJSYS::ASN1::RUNTIME::ASN1PERENCODETRACEHANDLER	45
COM::OBJSYS::ASN1::RUNTIME::ASN1PERINPUTSTREAM	46
COM::OBJSYS::ASN1::RUNTIME::ASN1PERMESSAGEBUFFER	47
COM::OBJSYS::ASN1::RUNTIME::ASN1PEROUTPUTSTREAM	48
COM::OBJSYS::ASN1::RUNTIME::ASN1PEROUTPUTSTREAMTRACEHANDLER	57
COM::OBJSYS::ASN1::RUNTIME::ASN1PERTRACEHANDLER	58
COM::OBJSYS::ASN1::RUNTIME::ASN1PERUTIL	60
COM::OBJSYS::ASN1::RUNTIME::ASN1SETDUPLICATEEXCEPTION	60
COM::OBJSYS::ASN1::RUNTIME::ASN1TAGGEDEVENTHANDLER	61
COM::OBJSYS::ASN1::RUNTIME::ASN1TAGMATCHFAILEDEXCEPTION	62
COM::OBJSYS::ASN1::RUNTIME::ASN1XERBASE64OCTETSTRING	63
COM::OBJSYS::ASN1::RUNTIME::ASN1XERDECODEBUFFER	65
COM::OBJSYS::ASN1::RUNTIME::ASN1XERELEMINFO	66
COM::OBJSYS::ASN1::RUNTIME::ASN1XERENCODEBUFFER	67
COM::OBJSYS::ASN1::RUNTIME::ASN1XERENCODER	73
COM::OBJSYS::ASN1::RUNTIME::ASN1XERENCODER_FIELDS	79
COM::OBJSYS::ASN1::RUNTIME::ASN1XEROPENTYPE	80
COM::OBJSYS::ASN1::RUNTIME::ASN1XEROPENTYPE::SAXHANDLER	82
COM::OBJSYS::ASN1::RUNTIME::ASN1XEROUTPUTSTREAM	83
COM::OBJSYS::ASN1::RUNTIME::ASN1XERSAXHANDLER	89
COM::OBJSYS::ASN1::RUNTIME::ASN1XERUTIL	92
COM::OBJSYS::ASN1::RUNTIME::ASN1XMLENCODEBUFFER	93
COM::OBJSYS::ASN1::RUNTIME::ASN1XMLOUTPUTSTREAM	94
COM::OBJSYS::ASN1::RUNTIME::ASN1XMLUTIL	95
COM::OBJSYS::ASN1::RUNTIME::XMLATTRIBUTES	96
COM::OBJSYS::ASN1::RUNTIME::XMLATTRIBUTES::XMLATTRIBUTE	101
COM::OBJSYS::ASN1::RUNTIME::XMLSAXCONTENTHANDLER	102
COM::OBJSYS::ASN1::RUNTIME::XMLSAXDEFAULTHANDLER	104
COM::OBJSYS::ASN1::RUNTIME::XMLSAXENTITYRESOLVER	107
COM::OBJSYS::ASN1::RUNTIME::XMLSAXERRORHANDLER	108
COM::OBJSYS::ASN1::RUNTIME::XMLSAXLEXICALHANDLER	108
COM::OBJSYS::ASN1::RUNTIME::XMLSAXLOCATOR	110

COM::OBJSYS::ASN1::RUNTIME::XMLSAXLOCATORIMPL.....	111
COM::OBJSYS::ASN1::RUNTIME::XMLSAXPARSER	113
COM::OBJSYS::ASN1::RUNTIME::XMLSAXPARSERADAPTER	117
COM::OBJSYS::ASN1::RUNTIME::XMLSOURCE.....	119
INDEX.....	121

ASN1C C# Runtime Library Part II Class Documentation

Asn1BerDecodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1BerDecodeBufferInherited by [Asn1BerInputStream](#), and [Asn1DerDecodeBuffer](#).

Detailed Description

This class handles the decoding of ASN.1 messages as specified in the Basic Encoding Rules (BER) as documented in the ITU-T X.690 standard.

Public Member Functions

- [Asn1BerDecodeBuffer](#) (System.IO.Stream istream)
- [Asn1BerDecodeBuffer](#) (byte[] msgdata)
- virtual int [DecodeLength](#) ()
- virtual byte[] [DecodeOpenType](#) (bool saveData)
- virtual byte[] [DecodeOpenType](#) ()
- virtual void [DecodeTag](#) (Asn1Tag tag)
- virtual int [DecodeTagAndLength](#) (Asn1Tag tag)
- virtual bool [MatchTag](#) (Asn1Tag tag)
- virtual bool [MatchTag](#) (Asn1Tag tag, Asn1Tag parsedTag, IntHolder parsedLen)
- virtual bool [MatchTag](#) (short tagClass, short tagForm, int tagIDCode, Asn1Tag parsedTag, IntHolder parsedLen)
- virtual void [Parse](#) ([Asn1TaggedEventHandler](#) handler)
- virtual Asn1Tag [PeekTag](#) ()
- virtual void [PeekTag](#) (Asn1Tag parsedTag)
- override int [ReadByte](#) ()

Static Public Member Functions

- static int [CalcIndefLen](#) (byte[] data, int offset, int len)

Protected Member Functions

- internal void [MovePastEOC](#) (bool saveData)

Properties

- virtual Asn1Tag [LastTag](#)
-

Constructor & Destructor Documentation

[Asn1BerDecodeBuffer](#) (byte[] *msgdata*)

This constructor creates a BER Decode buffer object that references an encoded ASN.1 message.

Parameters:

msgdata Byte array containing an encoded ASN.1 message.

[Asn1BerDecodeBuffer](#) (System.IO.Stream *istream*)

This constructor creates a BER Decode buffer object that references an encoded ASN.1 message. In this case, the message is passed in using an System.IO.Stream object.

Parameters:

istream Input stream containing an encoded ASN.1 message.

Member Function Documentation

static int CalcIndefLen (byte[] *data*, int *offset*, int *len*) [static]

This function calculates the actual length of an indefinite length message component.

Parameters:

data Buffer with the indefinite length message component.

offset The start offset in the array

len Length of the buffer (beginning from the offset)

Returns:

calculated length

virtual int DecodeLength () [virtual]

This method decodes a length value.

Returns:

Decoded length value

virtual byte [] DecodeOpenType (bool *saveData*) [virtual]

This method decodes an ASN.1 BER open type value. This is a fully encoded message component of any type. This version of the method allows the option of saving or discarding the open type data.

Parameters:

saveData True if data should be captured and returned

Returns:

Reference to byte array containing component.

virtual byte [] DecodeOpenType () [virtual]

This method decodes an ASN.1 BER open type value. This is a fully encoded message component of any type. The component is captured in the Decode capture buffer and a reference to a byte array is returned containing the component.

Returns:

Reference to byte array containing component.

virtual void DecodeTag (Asn1Tag tag) [virtual]

This method decodes a tag value.

Parameters:

tag Tag object to receive decoded tag fields.

Returns:

status value (see Asn1Status.java)

virtual int DecodeTagAndLength (Asn1Tag tag) [virtual]

This method decodes a tag and length value.

Parameters:

tag Tag object to receive decoded tag fields.

Returns:

Decoded length value.

virtual bool MatchTag (Asn1Tag tag) [virtual]

This overloaded version of MatchTag will just test for a match and not return parsed tag and length values

Parameters:

tag Tag value to be matched.

Returns:

True if given tag matches tag at Decode cursor

virtual bool MatchTag (Asn1Tag tag, Asn1Tag parsedTag, IntHolder parsedLen) [virtual]

This overloaded version of MatchTag allows the tag value to be matched to be passed using an Asn1Tag object.

Parameters:

tag Tag value to be matched.

parsedTag Holder object to receive parsed tag value

parsedLen Holder object to receive parsed length value

Returns:

True if given tag matches tag at Decode cursor

virtual bool MatchTag (short tagClass, short tagForm, int tagIDCode, Asn1Tag parsedTag, IntHolder parsedLen) [virtual]

This method decodes the next tag value and checks for a match with the given tag value. If the match is successful, the Decode cursor will be positioned at the contents field; otherwise, it will be reset to point to the start of the tag field.

Parameters:

tagClass Class value of tag to match

tagForm Form value of tag to match

tagIDCode ID code of tag to match

parsedTag Holder object to receive parsed tag value

parsedLen Holder object to receive parsed length value

Returns:

True if given tag matches tag at Decode cursor

internal void MovePastEOC (bool *saveData*) [protected]

This method skips or saves the data/bytes of the current tag in this DecodeBuffer. If current tag has indefinite length, than index will moved to end of the indifinite length. If tag has definite length, than that many bytes are moved.

Parameters:

saveData True if data should be captured

virtual void Parse ([Asn1TaggedEventHandler](#) *handler*) [virtual]

This method parses the complete message and invokes the event handler callback methods as various items are encountered.

Parameters:

handler Object implementing the Asn1EventHandler interface.

Returns:

Status value

virtual Asn1Tag PeekTag () [virtual]

This overloaded version of the PeekTag method will return a reference to a newly created tag object.

Returns:

Parsed tag object value reference

virtual void PeekTag (Asn1Tag *parsedTag*) [virtual]

This method will Parse and return the next tag in the Decode stream without advancing the Decode cursor.

Parameters:

parsedTag Holder object to receive parsed tag value

override int ReadByte ()

This 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

Property Documentation

virtual Asn1Tag LastTag [get]

Gets the last tag parsed within this decode buffer object.

Value: Last parsed tag object reference

Asn1BerDecodeContext Class Reference

Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext

Detailed Description

This class is mainly for internal use by the compiler to keep track of where nested constructed elements (SEQUENCE, SET, CHOICE, etc.) begin and end.

Public Member Functions

- [Asn1BerDecodeContext](#) ([Asn1BerDecodeBuffer](#) decodeBuffer, int elemLength)
- virtual bool [Expired](#) ()
- virtual bool [MatchElemTag](#) (Asn1Tag tag, IntHolder parsedLen, bool advance)
- virtual bool [MatchElemTag](#) (short tagClass, short tagForm, int tagIDCode, IntHolder parsedLen, bool advance)

Protected Attributes

- internal int [mDecBufByteCount](#)
- internal [Asn1BerDecodeBuffer](#) [mDecodeBuffer](#)
- internal int [mElemLength](#)
- internal bool [mExplicitTagging](#)
- internal Asn1Tag [mTagHolder](#)

Constructor & Destructor Documentation

[Asn1BerDecodeContext](#) ([Asn1BerDecodeBuffer](#) *decodeBuffer*, int *elemLength*)

The constructor initializes all internal working variables.

Parameters:

decodeBuffer Reference to current Decode buffer method.
elemLength Length of the element being tracked.

Member Function Documentation

virtual bool [Expired](#) () [virtual]

This method will determine if a decoding context is expired. A context is defined to be the wrapper in which a set of elements or a primitive data type resides..

Returns:

True if at the end of the context block

virtual bool [MatchElemTag](#) (Asn1Tag *tag*, IntHolder *parsedLen*, bool *advance*) [virtual]

This method will attempt to match the next element tag in a constructed type with the expected value. It will check to see if the context is expired and, if not, will match the given

tag with the expected tag. The Decode cursor is advanced if the boolean advance argument is true.

Parameters:

tag Tag object representing tag to be matched.
parsedLen Holder object to receive parsed length value
advance True if Decode cursor to be advanced.

Returns:

True, if the tag is matched

virtual bool MatchElemTag (short *tagClass*, short *tagForm*, int *tagIDCode*, IntHolder *parsedLen*, bool *advance*) [virtual]

This method will attempt to match the next element tag in a constructed type with the expected value. It will check to see if the context is expired and, if not, will match the given tag with the expected tag. The Decode cursor is advanced if the boolean advance argument is true.

Parameters:

tagClass Class value of tag to match
tagForm Form value of tag to match
tagIDCode ID code of tag to match
parsedLen Holder object to receive parsed length value
advance True if Decode cursor to be advanced.

Returns:

True, if the tag is matched

Member Data Documentation

internal int [mDecBufByteCount](#) [protected]

This variable is used to keep track of the current byte count in the Decode buffer.

internal [Asn1BerDecodeBuffer mDecodeBuffer](#) [protected]

This variable holds a reference to the BER Decode buffer object that is being used to Decode the entire message component.

internal int [mElemLength](#) [protected]

This variable holds the constructed element length for the context component.

internal bool [mExplicitTagging](#) [protected]

This boolean flag variable indicates if explicit tagging is in effect for this element.

internal Asn1Tag [mTagHolder](#) [protected]

This variable holds the current parsed tag for matching operations.

Asn1BerEncodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1BerEncodeBufferInherited by [Asn1DerEncodeBuffer](#).

Detailed Description

This class handles the encoding of ASN.1 messages as specified in the Basic Encoding Rules (BER) as specified in the ITU-T X.690 standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Public Member Functions

- [Asn1BerEncodeBuffer](#) (int sizeIncrement)
- [Asn1BerEncodeBuffer](#) ()
- virtual void [BinDump](#) ()
- override void [BinDump](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [Copy](#) (System.String data)
- virtual void [Copy](#) (byte[] data, int startOffset, int length)
- override void [Copy](#) (byte[] data)
- override void [Copy](#) (byte data)
- virtual int [EncodeIdentifier](#) (int ident)
- virtual int [EncodeIntValue](#) (long ivalue)
- virtual int [EncodeLength](#) (int len)
- virtual int [EncodeTag](#) (Asn1Tag tag)
- virtual int [EncodeTagAndLength](#) (short tagClass, short tagForm, int tagIDCode, int len)
- virtual int [EncodeTagAndLength](#) (Asn1Tag tag, int len)
- virtual int [EncodeUnsignedBinaryNumber](#) (long ivalue)
- override System.IO.Stream [GetInputStream](#) ()
- override void [Reset](#) ()
- override System.String [ToString](#) ()
- override void [Write](#) (System.IO.Stream outs)

Protected Member Functions

- internal override void [CheckSize](#) (int bytesRequired)

Properties

- virtual System.IO.MemoryStream [ByteArrayInputStream](#)
 - override byte[] [MsgCopy](#)
 - override int [MsgLength](#)
-

Constructor & Destructor Documentation

[Asn1BerEncodeBuffer](#) ()

This constructor creates a BER encode buffer object with the default size increment. Whenever the buffer becomes full, the buffer will be expanded by the sizeIncrement size.

Asn1BerEncodeBuffer (int *sizeIncrement*)

This constructor creates a BER encode buffer object with the given size increment. Whenever the buffer becomes full, the buffer will be expanded by the *sizeIncrement* size. This size should be large enough to prevent resizing in normal operation.

Parameters:

sizeIncrement The initial size in bytes of an encode buffer. If the buffer becomes full, it will be expanded by the amount.

Member Function Documentation

virtual void BinDump () [virtual]

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

override void BinDump (System.IO.StreamWriter *outs*, System.String *varName*)

This method dumps the encoded message in a human-readable format showing tags and contents to the given output stream.

Parameters:

outs StreamWriter where dump will be printed
varName Name of the Decoded ASN1 Type

internal override void CheckSize (int *bytesRequired*) [protected]

This method determines if the encode buffer can hold the requested number of bytes. If not, the buffer is expanded.

Parameters:

bytesRequired Number of required bytes.

virtual void Copy (System.String *data*) [virtual]

This method copies a character string into the encode buffer

Parameters:

data String to copy to the encode buffer

virtual void Copy (byte[] *data*, int *startOffset*, int *length*) [virtual]

This method copies multiple bytes to the encode buffer

Parameters:

data Array of bytes to copy to the encode buffer
startOffset The byte offset in array at which to begin copy.
length Number of bytes to copy

override void Copy (byte[] *data*)

This method copies multiple bytes to the encode buffer

Parameters:

data Array of bytes to copy to the encode buffer

override void Copy (byte *data*)

This method copies a single byte to the encode buffer.

Parameters:

data The byte value to copy

virtual int EncodeIdentifier (int *ident*) [virtual]

This method encodes an ASN.1 identifier value such as the ones used in a tags or object identifiers.

Parameters:

ident The identifier to be encoded.

Returns:

Length of the encoded component in octets.

virtual int EncodeIntValue (long *ivalue*) [virtual]

This method encodes an ASN.1 integer value's contents according to the ASN.1 Basic Encoding Rules (BER)..

Parameters:

ivalue Integer value to encode

Returns:

Length of encoded component

virtual int EncodeLength (int *len*) [virtual]

This method encodes a length value.

Parameters:

len The length to be encoded.

Returns:

Length of encoded component

virtual int EncodeTag (Asn1Tag *tag*) [virtual]

This method encodes a tag value.

Parameters:

tag The tag to be encoded.

Returns:

Length of component or negative status value

virtual int EncodeTagAndLength (short *tagClass*, short *tagForm*, int *tagIDCode*, int *len*) [virtual]

This overloaded version of encodeTagAndLength allows tag value components to be specified instead of an Asn1Tag object

Parameters:

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

len The length to be encoded.

Returns:

status value (see Asn1Status.java)

virtual int EncodeTagAndLength (Asn1Tag tag, int len) [virtual]

This method encodes both a tag and length value.

Parameters:

tag The tag to be encoded.

len The length to be encoded.

Returns:

Length of encoded component

virtual int EncodeUnsignedBinaryNumber (long ivalue) [virtual]

This method encodes an integer value as unsigned binary number according to the ASN.1 Basic Encoding Rules (BER)..

Parameters:

ivalue Integer value to encode

Returns:

Length of encoded component

override System.IO.Stream GetInputStream ()

This method returns an input stream representing the encoded message. This is a method defined as abstract in the base class that must be implemented by all derived classes. In this case, a byte array input stream is returned.

Returns:

Input stream containing encoded message

override void Reset ()

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

override System.String ToString ()

This method will return a string representation of the data in the encode buffer. The format is hex characters.

Returns:

Stringified representation of the value

override void Write (System.IO.Stream outs)

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

Parameters:

outs Output stream to which record is to be written

Property Documentation

virtual System.IO.MemoryStream ByteArrayInputStream [get]

This method returns a reference to a byte array input stream representing the encoded message. This is the preferred way to access the contents of the encoded message as it is the most efficient.

Returns:

byte array input stream containing encoded message

override byte [] MsgCopy [get]

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

Returns:

byte array containing encoded message

override int MsgLength [get]

This method returns the length of the encoded message component.

Returns:

length of encoded message component

Asn1BerInputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1BerInputStream inherits [Asn1BerDecodeBuffer](#).

Inherited by [Asn1CerInputStream](#).

Detailed Description

This class handles the input stream for the decoding of ASN.1 messages as specified in the Basic Encoding Rules (BER) as documented in the ITU-T X.690 standard.

Public Member Functions

- [Asn1BerInputStream](#) (System.IO.Stream istream)
 - virtual int [Available](#) ()
 - virtual void [Close](#) ()
 - override void [Mark](#) ()
 - virtual bool [MarkSupported](#) ()
 - override void [Reset](#) ()
 - override long [Skip](#) (long nbytes)
-

Constructor & Destructor Documentation

[Asn1BerInputStream](#) (System.IO.Stream *istream*)

This constructor creates a BER input stream object that references an encoded ASN.1 message.

Parameters:

istream Input stream containing an encoded ASN.1 message.

Member Function Documentation

virtual int Available () [virtual]

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.

virtual void Close () [virtual]

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

Exceptions:

System.SystemException if an I/O error occurs.

override 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.

virtual bool MarkSupported () [virtual]

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 .

override 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.

override 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

Asn1BerMessageDumpHandler Class Reference

Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandlerInherits
[Asn1TaggedEventHandler](#).

Detailed Description

This class implements the [Asn1EventHandler](#) interface to provide a formatted dump of a BER message to the given print output stream. An object of this type is used in conjunction with the [Asn1BerDecodeBuffer](#) *Parse* method to generically parse a BER message.

Public Member Functions

- [Asn1BerMessageDumpHandler](#) (System.IO.StreamWriter outs)
 - [Asn1BerMessageDumpHandler](#) ()
 - virtual void [Contents](#) (byte[] data)
 - virtual void [EndElement](#) (Asn1Tag tag)
 - virtual void [StartElement](#) (Asn1Tag tag, int len, byte[] tagLenBytes)
-

Constructor & Destructor Documentation

[Asn1BerMessageDumpHandler](#) ()

The constructor will print the dump result on the standard output stream.

[Asn1BerMessageDumpHandler](#) (System.IO.StreamWriter outs)

The constructor sets the StreamWriter object to which the formatted output should be written.

Parameters:

outs Output stream for formatted data

Member Function Documentation

virtual void [Contents](#) (byte[] *data*) [virtual]

This method is invoked after each contents field is parsed. It formats and prints the contents in a hex/ascii format.

Parameters:

data Array containing the encoded contents bytes

Implements [Asn1TaggedEventHandler](#).virtual void EndElement (Asn1Tag tag) [virtual]

This method is invoked after parsing is complete on each tag/length/value (TLV) in the message.

Parameters:

tag Array containing the encoded contents bytes

Implements [Asn1TaggedEventHandler](#).virtual void StartElement (Asn1Tag tag, int len, byte[] tagLenBytes) [virtual]

This method is invoked after each tag/length value is parsed in the message being dumped. It formats and prints the tag/length values.

Parameters:

tag Parsed tag value

len Parsed length value

tagLenBytes Array containing the encoded tag/length bytes

Implements [Asn1TaggedEventHandler](#).

Asn1BerOutputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1BerOutputStreamInherited by [Asn1CerOutputStream](#).

Detailed Description

This class implements the output stream to encode ASN.1 messages as specified in the Basic Encoding Rules (BER) as specified in the ITU-T X.690 standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Public Member Functions

- [Asn1BerOutputStream](#) (System.IO.Stream os, int bufSize)
- [Asn1BerOutputStream](#) (System.IO.Stream os)
- virtual void [Encode](#) (Asn1Type type, bool explicitTagging)
- virtual void [EncodeBitString](#) (byte[] data, int numbits, bool explicitTagging, Asn1Tag tag)
- virtual void [EncodeBMPString](#) (System.String data, bool explicitTagging, Asn1Tag tag)
- virtual void [EncodeCharString](#) (System.String data, bool explicitTagging, Asn1Tag tag)
- virtual void [EncodeEOC](#) ()
- virtual void [EncodeIdentifier](#) (long ident)
- virtual void [EncodeIntValue](#) (long data, bool encodeLen)
- virtual void [EncodeLength](#) (int len)
- virtual void [EncodeOctetString](#) (byte[] data, bool explicitTagging, Asn1Tag tag)
- virtual void [EncodeTag](#) (short tagClass, short tagForm, int tagIDCode)
- virtual void [EncodeTag](#) (Asn1Tag tag)
- virtual void [EncodeTagAndIndefLen](#) (short tagClass, short tagForm, int tagIDCode)
- virtual void [EncodeTagAndIndefLen](#) (Asn1Tag tag)
- virtual void [EncodeTagAndLength](#) (Asn1Tag tag, int len)
- virtual void [EncodeUnivString](#) (int[] data, bool explicitTagging, Asn1Tag tag)

- virtual void [EncodeUnsignedBinaryNumber](#) (long data)
-

Constructor & Destructor Documentation

[Asn1BerOutputStream](#) (System.IO.Stream os)

This constructor creates a buffered BER output stream object with default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

[Asn1BerOutputStream](#) (System.IO.Stream os, int bufSize)

This constructor creates a buffered BER output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

bufSize The buffer size. If it is 0 then the output stream is used as unbuffered one.

Member Function Documentation

virtual void Encode (Asn1Type type, bool explicitTagging) [virtual]

This method encodes and writes to the stream ASN.1 types. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified (the universal identifier must be provided by the caller).

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

Parameters:

type The object to be written

explicitTagging Flag indicating explicit tagging should be done

Reimplemented in [Asn1CerOutputStream](#).virtual void EncodeBitString (byte[] data, int numbits, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given array of bytes as bit string value.

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

Parameters:

data Byte array containing data to encode.

numbits Number of bits to encode

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented in [Asn1CerOutputStream](#).virtual void EncodeBMPString (System.String data, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given string as BMP string value.

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

Parameters:

data String containing data to encode.
explicitTagging Flag indicating explicit tagging should be done
tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented in [Asn1CerOutputStream](#).**virtual void EncodeCharString (System.String data, bool explicitTagging, Asn1Tag tag) [virtual]**

This method encodes and writes to the stream an 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).

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

Parameters:

data The string object to be written
explicitTagging Flag indicating explicit tagging should be done
tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented in [Asn1CerOutputStream](#).**virtual void EncodeEOC () [virtual]**

This method encodes and writes an End-Of-Contents marker to the stream.

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

virtual void EncodeIdentifier (long ident) [virtual]

This method encodes and writes to the stream an ASN.1 identifier value such as the ones used in a tags or object identifiers.

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

Parameters:

ident The identifier to be encoded.

virtual void EncodeIntValue (long data, bool encodeLen) [virtual]

This method encodes and writes to the stream an ASN.1 integer value's contents according to the ASN.1 Basic Encoding Rules (BER).

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

Parameters:

data Integer value to encode.
encodeLen Flag indicating length determinant should be encoded before encoding integer value.

virtual void EncodeLength (int len) [virtual]

This method encodes and writes a length value to the stream.

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

Parameters:

len The length to be encoded.

virtual void EncodeOctetString (byte[] data, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given array of bytes as octet string value.

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

Parameters:

data Byte array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented in [Asn1CerOutputStream](#).virtual void EncodeTag (short tagClass, short tagForm, int tagIDCode) [virtual]

This method encodes and writes a tag value to the stream.

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

Parameters:

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

virtual void EncodeTag (Asn1Tag tag) [virtual]

This method encodes and writes a tag value to the stream.

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

Parameters:

tag The tag to be encoded.

virtual void EncodeTagAndIndefLen (short tagClass, short tagForm, int tagIDCode) [virtual]

This overloaded version of EncodeTagAndIndefLen allows tag value components to be specified instead of an Asn1Tag object.

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

Parameters:

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

virtual void EncodeTagAndIndefLen (Asn1Tag tag) [virtual]

This method encodes and writes both a tag and an indefinite length indicator to the stream.

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

Parameters:

tag The tag to be encoded.

virtual void EncodeTagAndLength (Asn1Tag tag, int len) [virtual]

This method encodes and writes both a tag and length value to the stream.

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

Parameters:

tag The tag to be encoded.
len The length to be encoded.

virtual void EncodeUnivString (int[] data, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given array of integers as UniversalString value.
Throws, exception thrown by the underlying System.IO.Stream object.

Parameters:

data Array containing data to encode.
explicitTagging Flag indicating explicit tagging should be done
tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented in [Asn1CerOutputStream](#). **virtual void EncodeUnsignedBinaryNumber (long data) [virtual]**

This method encodes an integer value as unsigned binary number according to the ASN.1 Basic Encoding Rules (BER).. and writes to the stream
Throws, exception thrown by the underlying System.IO.Stream object.

Parameters:

data Integer value to encode.

Asn1CerInputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1CerInputStreamInherits [Asn1BerInputStream](#).

Detailed Description

This class handles the input stream for the decoding of ASN.1 messages as specified in the Canonical Encoding Rules (CER) as documented in the ITU-T X.690 standard.

Public Member Functions

- [Asn1CerInputStream](#) (System.IO.Stream istream)

Constructor & Destructor Documentation

[Asn1CerInputStream](#) (System.IO.Stream istream)

This constructor creates a CER input stream object that references an encoded ASN.1 message.

Parameters:

istream Input stream containing an encoded ASN.1 message.

Asn1CerOutputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1CerOutputStreamInherits [Asn1BerOutputStream](#).

Detailed Description

This class implements the output stream to encode ASN.1 messages as specified in the Canonical Encoding Rules (CER) as specified in the ITU-T X.690 standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Public Member Functions

- [Asn1CerOutputStream](#) (System.IO.Stream os, int bufSize)
- [Asn1CerOutputStream](#) (System.IO.Stream os)
- override void [Encode](#) (Asn1Type type, bool explicitTagging)
- override void [EncodeBitString](#) (byte[] value, int numbits, bool explicitTagging, Asn1Tag tag)
- override void [EncodeBMPString](#) (System.String value, bool explicitTagging, Asn1Tag tag)
- override void [EncodeCharString](#) (System.String value, bool explicitTagging, Asn1Tag tag)
- override void [EncodeOctetString](#) (byte[] value, bool explicitTagging, Asn1Tag tag)
- virtual void [EncodeStringTag](#) (int nbytes, short tagClass, short tagForm, int tagIDCode)
- virtual void [EncodeStringTag](#) (int nbytes, Asn1Tag tag)
- override void [EncodeUnivString](#) (int[] value, bool explicitTagging, Asn1Tag tag)

Constructor & Destructor Documentation

[Asn1CerOutputStream](#) (System.IO.Stream os)

This constructor creates a buffered CER output stream object with default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

[Asn1CerOutputStream](#) (System.IO.Stream os, int bufSize)

This constructor creates a buffered CER output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

bufSize The buffer size. If it is 0 then the output stream is used as unbuffered one.

Member Function Documentation

override void Encode (Asn1Type type, bool explicitTagging) [virtual]

This method encodes and writes to the stream ASN.1 types. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified (the universal identifier must be provided by the caller).

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

Parameters:

type The object to be written
explicitTagging Flag indicating explicit tagging should be done

Reimplemented from [Asn1BerOutputStream](#).override void EncodeBitString (byte[] value, int numbits, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given array of bytes as bit string value.

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

Parameters:

value Byte array containing data to encode.
numbits Number of bits to encode
explicitTagging Flag indicating explicit tagging should be done
tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1BerOutputStream](#).override void EncodeBMPString (System.String value, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given string as BMP string value.

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

Parameters:

value String containing data to encode.
explicitTagging Flag indicating explicit tagging should be done
tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1BerOutputStream](#).override void EncodeCharString (System.String value, bool explicitTagging, Asn1Tag tag) [virtual]

This method encodes and writes to the stream an 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).

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

Parameters:

value The string object to be written
explicitTagging Flag indicating explicit tagging should be done
tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1BerOutputStream](#). **override void EncodeOctetString (byte[] value, bool explicitTagging, Asn1Tag tag) [virtual]**

This method writes the given array of bytes as octet string value.

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

Parameters:

value Byte array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1BerOutputStream](#). **virtual void EncodeStringTag (int nbytes, short tagClass, short tagForm, int tagIDCode) [virtual]**

This method encodes and writes both a tag and length value to the stream.

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

Parameters:

nbytes The number of bytes in string to be encoded.

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

virtual void EncodeStringTag (int nbytes, Asn1Tag tag) [virtual]

This method encodes and writes both a tag and length value to the stream.

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

Parameters:

nbytes The number of bytes in string to be encoded.

tag The tag to be encoded.

override void EncodeUnivString (int[] value, bool explicitTagging, Asn1Tag tag) [virtual]

This method writes the given array of integers as UniversalString value.

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

Parameters:

value Array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1BerOutputStream](#).

Asn1DerDecodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1DerDecodeBufferInherits [Asn1BerDecodeBuffer](#).

Inherited by [Asn1DerInputStream](#).

Detailed Description

This class handles the decoding of ASN.1 messages as specified in the Distinguished Encoding Rules (DER) as documented in the ITU-T X.690 standard.

Public Member Functions

- [Asn1DerDecodeBuffer](#) (System.IO.Stream istream)
 - [Asn1DerDecodeBuffer](#) (byte[] msgdata)
-

Constructor & Destructor Documentation

[Asn1DerDecodeBuffer](#) (byte[] *msgdata*)

This constructor creates a DER Decode buffer object that references an encoded ASN.1 message.

Parameters:

msgdata Byte array containing an encoded ASN.1 message.

[Asn1DerDecodeBuffer](#) (System.IO.Stream *istream*)

This constructor creates a DER Decode buffer object that references an encoded ASN.1 message. In this case, the message is passed in using an System.IO.Stream object.

Parameters:

istream Input stream containing an encoded ASN.1 message.

Asn1DerEncodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1DerEncodeBufferInherits [Asn1BerEncodeBuffer](#).

Detailed Description

This class handles the encoding of ASN.1 messages as specified in the Distinguished Encoding Rules (DER) as specified in the ITU-T X.690 standard.

Public Member Functions

- [Asn1DerEncodeBuffer](#) (int sizeIncrement)
 - [Asn1DerEncodeBuffer](#) ()
-

Constructor & Destructor Documentation

[Asn1DerEncodeBuffer](#) ()

This constructor creates a DER encode buffer object with the default size increment. Whenever the buffer becomes full, the buffer will be expanded by the sizeIncrement size.

[Asn1DerEncodeBuffer](#) (int *sizeIncrement*)

This constructor creates a DER encode buffer object with the given size increment. Whenever the buffer becomes full, the buffer will be expanded by the sizeIncrement size. This size should be large enough to prevent resizing in normal operation.

Parameters:

sizeIncrement The initial size in bytes of an encode buffer. If the buffer becomes full, it will be expanded by this amount.

Asn1DerInputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1DerInputStream inherits [Asn1DerDecodeBuffer](#).

Detailed Description

This class handles the input stream for the decoding of ASN.1 messages as specified in the Distinguished Encoding Rules (DER) as documented in the ITU-T X.690 standard.

Public Member Functions

- [Asn1DerInputStream](#) (System.IO.Stream istream)
 - virtual int [Available](#) ()
 - virtual void [Close](#) ()
 - override void [Mark](#) ()
 - virtual bool [MarkSupported](#) ()
 - override void [Reset](#) ()
 - override long [Skip](#) (long nbytes)
-

Constructor & Destructor Documentation

[Asn1DerInputStream](#) (System.IO.Stream *istream*)

This constructor creates a DER decode buffer object that references an encoded ASN.1 message. In this case, the message is passed in using an System.IO.Stream object.

Parameters:

istream Input stream containing an encoded ASN.1 message.

Member Function Documentation**virtual int Available () [virtual]**

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.

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

Returns:

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

virtual void Close () [virtual]

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

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

override 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.

virtual bool MarkSupported () [virtual]

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 .

override 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.

override 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

Returns:

Skipped number of bytes

Asn1NotInSetException Class Reference

Com::Objsys::Asn1::Runtime::Asn1NotInSetException

Detailed Description

This class defines the 'ASN.1 element not in set' exception that is thrown from BER/DER methods when an element is parsed within the context of a SET that does not belong to the set.

Public Member Functions

- [Asn1NotInSetException](#) ([Asn1BerDecodeBuffer](#) buffer, Asn1Tag tag)
-

Constructor & Destructor Documentation

[Asn1NotInSetException](#) ([Asn1BerDecodeBuffer](#) *buffer*, Asn1Tag *tag*)

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

Parameters:

buffer BER decode buffer object reference
tag Tag value of element that is not in the set

Asn1PerBitField Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerBitField

Detailed Description

This class is used to store information on an individual bit field within a PER message. The information can be used to print a bit trace of the components of a message. It is used in conjunction with the [Asn1PerBitFieldList](#) class to map all bits in a message.

Public Member Functions

- [Asn1PerBitField](#) (System.String name, int bitOffset, int bitCount)
- virtual void [SetBitCountAndOffset](#) (int count, int offset)

Properties

- virtual int [BitCount](#)
 - virtual int [BitOffset](#)
 - virtual System.String [Name](#)
-

Constructor & Destructor Documentation

[Asn1PerBitField](#) (System.String *name*, int *bitOffset*, int *bitCount*)

This constructor initializes all of the variables used to track the bit fields.

Parameters:

name Name of the bit field.
bitOffset Offset within buffer to the bit field
bitCount Number of bits in the bit field

Member Function Documentation**virtual void SetBitCountAndOffset (int *count*, int *offset*) [virtual]**

This method sets the count of bits in the bit field and the offset to the bit field in the message buffer.

Parameters:

count Number of bits in the bit field
offset Offset within buffer to the bit field

Property Documentation**virtual int BitCount [get, set]**

Gets and Sets the number of bits in the bit field.

Value: Number of bits.

virtual int BitOffset [get, set]

Gets and Sets the offset to the bit field in the message buffer.

Value: Offset of the bitfield

virtual System.String Name [get]

This method returns the name assigned to the bit field.

Value: Bitfield name

Asn1PerBitFieldList Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList

Detailed Description

This class is used to map all of the bit fields in a PER message. After encoding or decoding is complete, this object can be used to provide a formatted printout of all of the message fields.

Public Member Functions

- virtual void [AddElemName](#) (System.String name, int arrayx)
- virtual System.Collections.IEnumerator [Iterator](#) ()

- virtual [Asn1PerBitField NewBitField](#) (System.String nameSuffix, int bitOffset, int bitCount)
- virtual void [RemoveLastElemName](#) ()
- virtual void [Reset](#) ()

Properties

- virtual int [BitOffset](#)
- virtual [Asn1PerBitField CurrBitField](#)

Member Function Documentation

virtual void [AddElemName](#) (System.String *name*, int *arrayx*) [**virtual**]

This method adds an element name to the current fully qualified name. The fully qualified name is a string of name components separated by dots (ex. a.b.c).

Parameters:

name Name component to append to string
arrayx Array index if named item is an element in an array (set to -1 otherwise)

virtual System.Collections.IEnumerator [Iterator](#) () [**virtual**]

This method returns an iterator to the encapsulated bit field linked list object.

Returns:

System.Collections.IEnumerator value of this list

virtual [Asn1PerBitField NewBitField](#) (System.String *nameSuffix*, int *bitOffset*, int *bitCount*) [**virtual**]

This method creates a new bit field object with the given properties and appends it to the bit field list. Also sets as current bit field.

Parameters:

nameSuffix Suffix to add to fully qualified name for this field (for example, 'length')
bitOffset Offset to the start of this field in bits from the beginning of the encode buffer.
bitCount Number of bits in the field.

Returns:

Created bit field

virtual void [RemoveLastElemName](#) () [**virtual**]

This method removes the last element name in the current fully qualified name string. For example, if the current string is 'a.b.c', it will be 'a.b' after calling this method.

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

This method resets the object.

Property Documentation

virtual int [BitOffset](#) [**set**]

Set the current bit offset in the bit field.

Value: The bit offset

virtual [Asn1PerBitField](#) CurrBitField [get]

Gets a reference to the current bit field object (i.e. the one that was last created).

Value: Current bit field

Asn1PerBitFieldPrinter Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter

Detailed Description

This class is used to obtain a formatted printout of the bit fields that make up a PER encoded message.

Public Member Functions

- [Asn1PerBitFieldPrinter](#) ([Asn1PerMessageBuffer](#) perMessageBuffer, System.IO.Stream encodedMessage)
- virtual void [Print](#) (System.IO.StreamWriter outs, System.String varName)

Protected Attributes

- internal int [mBitMask](#)
 - internal int [mByteIndex](#)
 - internal int [mCurrOctet](#)
 - internal System.IO.Stream [mEncodedMessage](#)
 - internal int [mFmtAscCharIdx](#)
 - internal int [mFmtBitCharIdx](#)
 - internal int [mFmtHexCharIdx](#)
 - internal System.Text.StringBuilder [mFormatBuffer](#)
 - internal [Asn1PerMessageBuffer](#) [mPerMessageBuffer](#)
-

Constructor & Destructor Documentation

[Asn1PerBitFieldPrinter](#) ([Asn1PerMessageBuffer](#) perMessageBuffer, System.IO.Stream encodedMessage)

Constructor

Parameters:

perMessageBuffer PER encode or decode message buffer

encodedMessage Input stream of encoded message

Member Function Documentation

virtual void Print (System.IO.StreamWriter *outs*, System.String *varName*) [virtual]

This method iterates through and prints all of the bit fields in a PER encoded message. Bit tracing needs to have been enabled in the buffer via the 'perTraceEnable' method prior to encoding or decoding the message.

Parameters:

outs Print stream

varName Variable name. This will be printed before all fields (for example, <varName> .field1, etc.)

Member Data Documentation

internal int [mBitMask](#) [protected]

This variable holds the mask for current bit

internal int [mByteIndex](#) [protected]

This variable holds the byte index

internal int [mCurrOctet](#) [protected]

This variable holds the current byte

internal System.IO.Stream [mEncodedMessage](#) [protected]

This variable holds the input stream

internal int [mFmtAscCharIdx](#) [protected]

This variable holds the index for formatted information

internal int [mFmtBitCharIdx](#) [protected]

This variable holds the index for formatted information

internal int [mFmtHexCharIdx](#) [protected]

This variable holds the index for formatted information

internal System.Text.StringBuilder [mFormatBuffer](#) [protected]

```
Initial value:  
new System.Text.StringBuilder()
```

This variable holds the formatted information of current byte

internal [Asn1PerMessageBuffer](#) [mPerMessageBuffer](#) [protected]

This variable holds the PER encode or decode message buffer

Asn1PerDecodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerDecodeBufferInherits [Asn1PerMessageBuffer](#).

Inherited by [Asn1PerInputStream](#).

Detailed Description

This class handles the decoding of ASN.1 messages as specified in the Packed Encoding Rules (PER) ITU-T X.691 standard.

Public Member Functions

- [Asn1PerDecodeBuffer](#) (System.IO.Stream istream, bool aligned)
- [Asn1PerDecodeBuffer](#) (byte[] msgdata, bool aligned)
- virtual void [BinDump](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [BinDump](#) (System.String varName)
- virtual void [ByteAlign](#) ()
- virtual bool [DecodeBit](#) ()
- virtual bool [DecodeBit](#) (System.String ident)
- virtual int [DecodeBitsToInt](#) (int nbits)
- virtual int [DecodeBitsToInt](#) (int nbits, System.String ident)
- virtual long [DecodeBitsToLong](#) (int nbits)
- virtual long [DecodeBitsToLong](#) (int nbits, System.String ident)
- virtual void [DecodeBitsToOctetArray](#) (byte[] data, int offset, int nbits)
- virtual void [DecodeBitsToOctetArray](#) (byte[] data, int offset, int nbits, System.String ident)
- virtual void [DecodeCharString](#) (int nchars, int abpc, int ubpc, Asn1CharSet charSet, System.Text.StringBuilder sbuf)
- virtual long [DecodeConsWholeNumber](#) (long rangeValue)
- virtual long [DecodeConsWholeNumber](#) (long rangeValue, System.String ident)
- virtual long [DecodeExtLength](#) ()
- virtual long [DecodeInt](#) (int nocts, bool signExtend)
- virtual long [DecodeInt](#) (int nocts, bool signExtend, System.String ident)
- virtual long [DecodeLength](#) (long lower, long upper)
- virtual long [DecodeLength](#) ()
- virtual int [DecodeSmallNonNegWholeNumber](#) ()
- virtual bool [IsAligned](#) ()
- virtual void [MoveBitCursor](#) (long offset)
- override int [ReadByte](#) ()
- virtual void [SetAligned](#) (bool data)
- override void [SetInputStream](#) (byte[] msgdata, int offset, int length)

Static Public Member Functions

- static [Asn1PerDecodeBuffer SetBuffer](#) ([Asn1PerDecodeBuffer](#) buffer, byte[] msgdata, bool aligned)

Properties

- virtual long [BitOffset](#)
- virtual int [MsgBitCnt](#)
- internal [Asn1PerTraceHandler mTraceHandler](#)
- virtual [Asn1PerTraceHandler TraceHandler](#)

Constructor & Destructor Documentation

[Asn1PerDecodeBuffer](#) (byte[] *msgdata*, bool *aligned*)

This constructor creates a PER Decode buffer object that references an encoded ASN.1 message.

Parameters:

msgdata Byte array containing an encoded ASN.1 message.
aligned `true` for specifying PER aligned; otherwise `false` for unaligned encoding.

[Asn1PerDecodeBuffer](#) (System.IO.Stream *istream*, bool *aligned*)

This constructor creates a PER Decode buffer object that references an encoded ASN.1 message. In this case, the message is passed in using an System.IO.Stream object.

Parameters:

istream Input stream containing an encoded ASN.1 message.
aligned Boolean specifying PER aligned or unaligned encoding.

Member Function Documentation

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

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

Parameters:

outs StreamWriter object to which output should be written
varName Name of top-level message object variable

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 top-level message object variable

virtual void ByteAlign () [virtual]

This methods byte-aligns the buffer.

Implements [Asn1PerMessageBuffer](#).virtual bool DecodeBit () [virtual]

This method decodes a single bit value. The `ident` argument which is used for tracing is defaulted to 'value'.

Returns:

Boolean value of bit that was decoded.

virtual bool DecodeBit (System.String *ident*) [virtual]

This method decodes a single bit value.

Parameters:

ident Bit field identifier name for tracing.

Returns:

Boolean value of bit that was decoded.

virtual int DecodeBitsToInt (int *nbits*) [virtual]

This method decodes bits from the input stream into a standard integer value. Up to 32 bits can be decoded. The bits are placed in the least-significant bytes of the integer. The *ident* argument which is used for tracing is defaulted to 'value'.

Returns:

Integer value containing decoded bits

Parameters:

nbits Number of bits to Decode

virtual int DecodeBitsToInt (int *nbits*, System.String *ident*) [virtual]

This method decodes bits from the input stream into a standard integer value. Up to 32 bits can be decoded. The bits are placed in the least-significant bytes of the integer.

Parameters:

nbits Number of bits to Decode

ident Bit field identifier name for tracing.

Returns:

Integer value containing decoded bits

virtual long DecodeBitsToLong (int *nbits*) [virtual]

This method decodes bits from the input stream into a long integer value. Up to 64 bits can be decoded. The bits are placed in the least-significant bytes of the long integer. The *ident* argument which is used for tracing is defaulted to 'value'.

Parameters:

nbits Number of bits to Decode

Returns:

Long integer value containing decoded bits

virtual long DecodeBitsToLong (int *nbits*, System.String *ident*) [virtual]

This method decodes bits from the input stream into a long integer value. Up to 64 bits can be decoded. The bits are placed in the least-significant bytes of the long integer.

Returns:

Long integer value containing decoded bits

Parameters:

nbits Number of bits to Decode

ident Bit field identifier name for tracing.

virtual void DecodeBitsToOctetArray (byte[] *data*, int *offset*, int *nbits*) [virtual]

This method decodes bits from the input stream into an array of octets. The user is expected to have provided an array large enough to hold the number of bits requested to be decoded. The *ident* argument which is used for tracing is defaulted to 'value'.

Parameters:

data Octet array for decoded data
offset Starting byte offset into array
nbits Number of bits to Decode

virtual void DecodeBitsToOctetArray (byte[] data, int offset, int nbits, System.String ident) [virtual]

This method decodes bits from the input stream into an array of octets. The user is expected to have provided an array large enough to hold the number of bits requested to be decoded.

Parameters:

data Octet array for decoded data
offset Starting byte offset into array
nbits Number of bits to Decode
ident Bit field identifier name for tracing.

virtual void DecodeCharString (int nchars, int abpc, int ubpc, Asn1CharSet charSet, System.Text.StringBuilder sbuf) [virtual]

This method decodes the contents of a known-multiplier character string. This version of the method assumes a permitted alphabet constraint is in place.

Parameters:

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)
sbuf String buffer to receive decoded result

virtual long DecodeConsWholeNumber (long rangeValue) [virtual]

This method implements the rules to Decode a constrained whole number as specified in section 10.5 of the X.691 standard. The *ident* argument which is used for tracing is defaulted to 'value'.

Parameters:

rangeValue lower - upper + 1

Returns:

Decoded adjusted value = value - lower range endpoint value

virtual long DecodeConsWholeNumber (long rangeValue, System.String ident) [virtual]

This method implements the rules to Decode a constrained whole number as specified in section 10.5 of the X.691 standard.

Parameters:

rangeValue lower - upper + 1
ident Tracing identifier

Returns:

Decoded adjusted value = value - lower range endpoint value

virtual long DecodeExtLength () [virtual]

This method decodes an extension length value. Note that the decoded length is not what is returned. The bit offset to the start of the next element within the Decode buffer is returned.

Returns:

Bit offset to next element in buffer

virtual long DecodeInt (int *nocts*, bool *signExtend*) [virtual]

This method implements the rules to Decode an unconstrained integer value. The *ident* argument which is used for tracing is defaulted to 'value'.

Returns:

Decoded long integer value

Parameters:

nocts Number of octets to Decode
signExtend Sign extend resulting value

virtual long DecodeInt (int *nocts*, bool *signExtend*, System.String *ident*) [virtual]

This method implements the rules to Decode an unconstrained integer value.

Returns:

Decoded long integer value

Parameters:

nocts Number of octets to Decode
signExtend Sign extend resulting value
ident Tracing identifier

virtual long DecodeLength (long *lower*, long *upper*) [virtual]

This method decodes a constrained length determinant value.

Parameters:

lower Lower bound (inclusive) of length value range
upper Upper bound (inclusive) of length value range

Returns:

Decoded length value

virtual long DecodeLength () [virtual]

This method decodes a general (unconstrained) length determinant value as described in section 10.9 of the X.691 standard. The maximum value that will be returned is 64K which is the largest length fragment size defined for PER. If a value of 16K or larger is returned, the user must repeat this call to fully Decode the fragmented contents.

Returns:

Decoded length value. If the returned value is $\geq 16k$, this indicates a fragmented length was decoded. The user must call this method again after the data fragment is decoded to get the next fragment length.

virtual int DecodeSmallNonNegWholeNumber () [virtual]

This method implements the rules to Decode a small non-negative whole number as specified in section 10.6 of the X.691 standard.

Returns:

Decoded int value

virtual bool IsAligned () [virtual]

This method tests if PER alignment is turned on or off.

Returns:

`true` for PER aligned encoding or `false` for unaligned encoding.

Implements [Asn1PerMessageBuffer](#).virtual void MoveBitCursor (long *offset*) [virtual]

This method moves the bit cursor to the given offset.

Parameters:

offset Absolute bit offset value

override int ReadByte ()

This 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

virtual void SetAligned (bool *data*) [virtual]

This method is used to turn PER aligned encoding on or off.

Parameters:

data `true` for PER aligned encoding or `false` for unaligned encoding.

static [Asn1PerDecodeBuffer](#) SetBuffer ([Asn1PerDecodeBuffer](#) *buffer*, byte[] *msgdata*, bool *aligned*) [static]

This method will create or reinitialize a PER Decode message buffer object to read data from the given byte array. If the existing buffer reference is null, a new buffer will be created; otherwise, the existing buffer will be reused.

Parameters:

buffer Existing message buffer object

msgdata Byte array containing message data

aligned `true` specifying PER aligned or `false` for unaligned encoding.

Returns:

[Asn1PerDecodeBuffer](#) to read data from the given byte array

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

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

Property Documentation

virtual long BitOffset [get]

Gets the absolute offset to the current bit in the Decode buffer.

Value: offset to current bit in Decode buffer

virtual int MsgBitCnt [get]

Gets the number of bits in the encoded PER message.

Value: count of bits in encoded message

Implements [Asn1PerMessageBuffer](#).internal [Asn1PerTraceHandler](#) mTraceHandler [protected]

Variable holds the PER message trace handler

virtual [Asn1PerTraceHandler](#) TraceHandler [get]

Gets a reference to the internal trace handler object used to trace the bit fields within a PER message.

Value: [Asn1PerTraceHandler](#) object

Implements [Asn1PerMessageBuffer](#).

Asn1PerDecodeTraceHandler Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandlerInherits [Asn1PerTraceHandler](#).

Detailed Description

This is a utility class for handling the collection and printing of PER bit field trace information. An object of the class is present within both the [Asn1PerEncodeBuffer](#) and [Asn1PerDecodeBuffer](#) classes. It is accessed using the 'TraceHandler' property from within objects of these classes.

Public Member Functions

- [Asn1PerDecodeTraceHandler](#) ([Asn1PerDecodeBuffer](#) messageBuffer)
 - override void [Enable](#) ()
 - override void [Print](#) (System.IO.StreamWriter outs, System.String varName)
 - override void [Reset](#) ()
-

Constructor & Destructor Documentation

[Asn1PerDecodeTraceHandler](#) ([Asn1PerDecodeBuffer](#) messageBuffer)

This constructor initializes the internal trace handler member variables.

Parameters:

messageBuffer PER decode message buffer object reference

Member Function Documentation

override void Enable () [virtual]

This method is used to turn PER bit tracing on

Implements [Asn1PerTraceHandler](#).override void Print (System.IO.StreamWriter outs, System.String varName) [virtual]

This method prints the trace to the given output stream in a default format.

Parameters:

outs Print stream to which output is to be written.
varName Name of the object variable being printed.

Implements [Asn1PerTraceHandler](#).override void Reset () [virtual]

This method resets the trace bit field list.

Implements [Asn1PerTraceHandler](#).

Asn1PerEncodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerEncodeBufferInherits [Asn1PerMessageBuffer](#).

Detailed Description

This class handles the encoding of ASN.1 messages as specified in the Packed Encoding Rules (PER) ITU-T X.691 standard.

Public Member Functions

- [Asn1PerEncodeBuffer](#) (bool aligned, int sizeIncrement)
- [Asn1PerEncodeBuffer](#) (bool aligned)
- override void [BinDump](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [ByteAlign](#) ()
- override void [Copy](#) (byte[] value)
- override void [Copy](#) (byte value)
- virtual void [EncodeBit](#) (bool value)
- virtual void [EncodeBit](#) (bool value, System.String ident)
- virtual void [EncodeBits](#) (byte[] value, int offset, int nbits)
- virtual void [EncodeBits](#) (byte[] value, int offset, int nbits, System.String ident)
- virtual void [EncodeBits](#) (byte value, int nbits)
- virtual void [EncodeCharString](#) (System.String value, int nchars, int offset, int abpc, int ubpc, Asn1CharSet charSet)
- virtual void [EncodeConsWholeNumber](#) (long adjustedValue, long rangeValue)
- virtual void [EncodeConsWholeNumber](#) (long adjustedValue, long rangeValue, System.String ident)
- virtual void [EncodeInt](#) (long value, bool encodeLen, bool signExtend)
- virtual void [EncodeInt](#) (long value, bool encodeLen, bool signExtend, System.String ident)
- virtual void [EncodeInt](#) (long value, int nbits)

- virtual void [EncodeInt](#) (long value, int nbits, System.String ident)
- virtual void [EncodeLength](#) (long value, long lower, long upper)
- virtual long [EncodeLength](#) (long value)
- virtual void [EncodeLengthEOM](#) (long value)
- virtual void [EncodeOctetString](#) (byte[] value, int offset, int nbytes)
- virtual void [EncodeOIDLengthAndValue](#) (int[] value)
- virtual void [EncodeOpenType](#) ([Asn1PerEncodeBuffer](#) buffer, System.String elemName)
- virtual void [EncodeOpenType](#) (byte[] value, int offset, int nbytes)
- virtual void [EncodeRelOIDLengthAndValue](#) (int[] value)
- virtual void [EncodeSmallNonNegWholeNumber](#) (int value)
- override System.IO.Stream [GetInputStream](#) ()
- override void [HexDump](#) ()
- virtual bool [IsAligned](#) ()
- override void [Reset](#) ()
- virtual void [ReverseBytes](#) (int offset, int nbytes)
- virtual void [SetAligned](#) (bool value)
- override System.String [ToString](#) ()
- override void [Write](#) (System.IO.Stream outs)

Protected Member Functions

- internal override void [CheckSize](#) (int bytesRequired)

Properties

- virtual byte[] [Buffer](#)
- virtual System.IO.MemoryStream [ByteArrayInputStream](#)
- virtual int [ByteIndex](#)
- virtual int [MsgBitCnt](#)
- virtual int [MsgByteCnt](#)
- override byte[] [MsgCopy](#)
- override int [MsgLength](#)
- internal [Asn1PerTraceHandler mTraceHandler](#)
- virtual [Asn1PerTraceHandler TraceHandler](#)

Constructor & Destructor Documentation

[Asn1PerEncodeBuffer](#) (bool *aligned*)

This constructor creates a PER encode buffer object with the default size increment. Whenever the buffer becomes full, the buffer will be expanded by the `sizeIncrement` size.

Parameters:

aligned `true` for PER aligned or `false` for PER unaligned encoding.

[Asn1PerEncodeBuffer](#) (bool *aligned*, int *sizeIncrement*)

This constructor creates a PER encode buffer object with the given size increment. Whenever the buffer becomes full, the buffer will be expanded by the `sizeIncrement` size. This size should be large enough to prevent resizing in normal operation.

Parameters:

aligned `true` for PER aligned or `false` for PER unaligned encoding.

sizeIncrement The initial size in bytes of an encode buffer. If the buffer becomes full, it will be expanded by the amount.

Member Function Documentation

override void BinDump (System.IO.StreamWriter *outs*, System.String *varName*)

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 StreamWriter object to which output should be written
varName Name of top-level message object variable

virtual void ByteAlign () [virtual]

This methods byte-aligns the buffer.

Implements [Asn1PerMessageBuffer](#).internal override void CheckSize (int *bytesRequired*) [protected]

This method determines if the encode buffer can hold the requested number of bytes. If not, the buffer is expanded.

Parameters:

bytesRequired Number of required bytes.

override void Copy (byte[] *value*)

This method copies multiple bytes to the encode buffer

Parameters:

value Array of bytes to copy to the encode buffer

override void Copy (byte *value*)

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

Parameters:

value The byte value to copy

virtual void EncodeBit (bool *value*) [virtual]

This method encodes a single bit value. The *ident* argument which is used for tracing is defaulted to 'value'.

Parameters:

value Boolean value of bit to be encoded.

virtual void EncodeBit (bool *value*, System.String *ident*) [virtual]

This method encodes a single bit value.

Parameters:

value Boolean value of bit to be encoded.
ident Bit field identifier name for tracing.

virtual void EncodeBits (byte[] value, int offset, int nbits) [virtual]

This method encodes bit values from an array of octets. The `ident` argument which is used for tracing is defaulted to 'value'.

Parameters:

value Octet array containing bits to be encoded
offset Starting byte offset in value
nbits Number of bits to encode

virtual void EncodeBits (byte[] value, int offset, int nbits, System.String ident) [virtual]

This method encodes bit values from an array of octets.

Parameters:

value Octet array containing bits to be encoded
offset Starting byte offset in value
nbits Number of bits to encode
ident Bit field identifier name for tracing.

virtual void EncodeBits (byte value, int nbits) [virtual]

This method encodes bit values from an octet. The most significant bits from the octet are encoded.

Parameters:

value Octet containing bits to be encoded
nbits Number of bits to encode

virtual void EncodeCharString (System.String value, int nchars, int offset, int abpc, int ubpc, Asn1CharSet charSet) [virtual]

This method encodes the contents of a known-multiplier character string type. This version assumes a permitted alphabet constraint was specified.

Parameters:

value String containing characters to encode
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)

virtual void EncodeConsWholeNumber (long adjustedValue, long rangeValue) [virtual]

This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard. The `ident` argument which is used for tracing is defaulted to 'value'.

Parameters:

adjustedValue Adjusted value to be encoded = value - lower range endpoint value
rangeValue lower - upper + 1

virtual void EncodeConsWholeNumber (long adjustedValue, long rangeValue, System.String ident) [virtual]

This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.

Parameters:

adjustedValue Adjusted value to be encoded = value - lower range endpoint value
rangeValue lower - upper + 1
ident Tracing identifier

virtual void EncodeInt (long value, bool encodeLen, bool signExtend) [virtual]

This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard. The *ident* argument which is used for tracing is defaulted to 'value'.

Parameters:

value Integer value to be encoded
encodeLen Flag indicating length determinant should be encoded before encoding integer value.
signExtend Flag indicating if sign extension should be performed.

virtual void EncodeInt (long value, bool encodeLen, bool signExtend, System.String ident) [virtual]

This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.

Parameters:

value Integer value to be encoded
encodeLen Flag indicating length determinant should be encoded before encoding integer value.
signExtend Flag indicating if sign extension should be performed.
ident Tracing identifier

virtual void EncodeInt (long value, int nbits) [virtual]

This method encodes bit values from an integer value. The least significant bits from the integer are encoded. The *ident* argument which is used for tracing is defaulted to 'value'.

Parameters:

value Integer containing bits to be encoded
nbits Number of bits to encode

virtual void EncodeInt (long value, int nbits, System.String ident) [virtual]

This method encodes bit values from an integer value. The least significant bits from the integer are encoded.

Parameters:

value Integer containing bits to be encoded
nbits Number of bits to encode
ident Tracing identifier

virtual void EncodeLength (long value, long lower, long upper) [virtual]

This method encodes a constrained length determinant value.

Parameters:

value Length value to be encoded
lower Lower bound (inclusive) of length value range
upper Upper bound (inclusive) of length value range

virtual long EncodeLength (long value) [virtual]

This method encodes a general (unconstrained) length determinant value as described in section 10.9 or the X.691 standard.

Parameters:

value Length value to be encoded

Returns:

Value that was actually encoded. This may be less than the value that was passed in if fragmentation was done (i.e the value was $\geq 16k$).

virtual void EncodeLengthEOM (long value) [virtual]

This method checks to see if a zero byte needs to be added after a fragmented length has been encoded. It will add it to the byte stream if necessary.

Parameters:

value Original length value that was encoded.

virtual void EncodeOctetString (byte[] value, int offset, int nbytes) [virtual]

This method encodes the given array of bytes as an unconstrained octet string value.

Parameters:

value Byte array containing data to encode. This is assumed to contain a previously encoded PER component.

offset Starting offset in byte array value

nbytes Number of bytes to encode

virtual void EncodeOIDLengthAndValue (int[] value) [virtual]

This method encodes the length and contents of an object identifier value.

Parameters:

value Integer array containing arcs to encode

virtual void EncodeOpenType ([Asn1PerEncodeBuffer](#) buffer, System.String elemName) [virtual]

This overloaded version of encodeOpenType will encode the component in the given PER encode buffer into this PER encode buffer.

Parameters:

buffer PER encode buffer containing encoded message component.

elemName Name of element being encoded.

virtual void EncodeOpenType (byte[] value, int offset, int nbytes) [virtual]

This method encodes the given array of bytes as an open type.

Parameters:

value Byte array containing data to encode. This is assumed to contain a previously encoded PER component.

offset Starting offset in byte array value

nbytes Number of bytes to encode

virtual void EncodeRelOIDLengthAndValue (int[] value) [virtual]

This method encodes the length and contents of a relative object identifier value.

Parameters:

value Integer array containing arcs to encode

virtual void EncodeSmallNonNegWholeNumber (int *value*) [virtual]

This method implements the rules to encode a small non-negative whole number as specified in section 10.6 of the X.691 standard.

Parameters:

value Value to be encoded

override System.IO.Stream GetInputStream ()

This method returns an input stream representing the encoded message. This method is defined as abstract in the base class and must be implemented by all derived classes. In this case, a byte array input stream is returned.

Returns:

Input stream containing encoded message

Implements [Asn1PerMessageBuffer.override void HexDump \(\)](#)

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

virtual bool IsAligned () [virtual]

This method is used to test if PER aligned encoding has been specified.

Returns:

true for PER aligned encoding or *false* for unaligned encoding.

Implements [Asn1PerMessageBuffer.override void Reset \(\)](#)

This method resets the buffer object so that it can be reused to encode another PER message. Any previously encoded data is lost.

virtual void ReverseBytes (int *offset*, int *nbytes*) [virtual]

This method reverses a series of bytes at a given offset within the encode buffer.

Parameters:

offset Starting byte offset within the buffer
nbytes Number of bytes to reverse

virtual void SetAligned (bool *value*) [virtual]

This method is used to turn PER aligned encoding on or off

Parameters:

value *true* for PER aligned encoding or *false* for unaligned encoding.

override System.String ToString ()

This method will return a string representation of the data in the encode buffer. The format is hex characters.

Returns:

Stringified representation of the value

override void Write (System.IO.Stream outs)

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

Parameters:

outs Output stream to which record is to be written

Property Documentation

virtual byte [] Buffer [get]

Gets a reference to the byte buffer used to hold the encoded message.

Value: Byte buffer reference

virtual System.IO.MemoryStream ByteArrayInputStream [get]

Gets a reference to a byte array input stream representing the encoded message. This is the preferred way to access the contents of the encoded message as it is the most efficient.

Value: byte array input stream containing encoded message

virtual int ByteIndex [get]

Gets the current byte index into the encode buffer.

Value: Byte index value

virtual int MsgBitCnt [get]

Gets the number of bits in the encoded PER message.

Value: Count of bits in encoded message

Implements [Asn1PerMessageBuffer](#).virtual int MsgByteCnt [get]

Gets the number of bytes in the encoded PER message. The number is rounded up to include the last byte if one or more bits have been set in that byte.

Value: Count of bytes in encoded message

override byte [] MsgCopy [get]

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

Value: byte array containing encoded message

override int MsgLength [get]

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

Value: length of encoded message component

internal [Asn1PerTraceHandler](#) mTraceHandler [protected]

Variable holds the PER message trace handler

virtual [Asn1PerTraceHandler](#) TraceHandler [get]

Gets a reference to the internal trace handler object used to trace the bit fields within a PER message.

Value: [Asn1PerTraceHandler](#) object

Implements [Asn1PerMessageBuffer](#).

Asn1PerEncodeTraceHandler Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandlerInherits [Asn1PerTraceHandler](#).

Detailed Description

This is a utility class for handling the collection and printing of PER bit field trace information. An object of the class is present within both the [Asn1PerEncodeBuffer](#) and [Asn1PerDecodeBuffer](#) classes. It is accessed using the 'TraceHandler' property from objects of these classes.

Public Member Functions

- [Asn1PerEncodeTraceHandler](#) ([Asn1PerEncodeBuffer](#) messageBuffer)
 - override void [Enable](#) ()
 - override void [Print](#) (System.IO.StreamWriter outs, System.String varName)
 - override void [Reset](#) ()
-

Constructor & Destructor Documentation

[Asn1PerEncodeTraceHandler](#) ([Asn1PerEncodeBuffer](#) messageBuffer)

This constructor initializes internal trace handler member variables.

Parameters:

messageBuffer PER encode message buffer object reference

Member Function Documentation

override void [Enable](#) () [virtual]

This method is used to turn PER bit tracing on

Implements [Asn1PerTraceHandler](#).override void [Print](#) (System.IO.StreamWriter *outs*, System.String *varName*) [virtual]

This method prints the trace to the given output stream in a default format.

Parameters:

outs Print stream to which output is to be written.

varName Name of the object variable being printed.

Implements [Asn1PerTraceHandler](#).override void **Reset ()** [virtual]

This method resets the trace bit field list.

Implements [Asn1PerTraceHandler](#).

Asn1PerInputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerInputStreamInherits [Asn1PerDecodeBuffer](#).

Detailed Description

This class handles the input stream for decoding of ASN.1 messages as specified in the Packed Encoding Rules (PER) ITU-T X.691 standard.

Public Member Functions

- [Asn1PerInputStream](#) (System.IO.Stream *istream*, bool *aligned*)
 - virtual int [Available](#) ()
 - virtual void [Close](#) ()
 - override void [Mark](#) ()
 - virtual bool [MarkSupported](#) ()
 - override void [Reset](#) ()
 - override long [Skip](#) (long *nbytes*)
-

Constructor & Destructor Documentation

[Asn1PerInputStream](#) (System.IO.Stream *istream*, bool *aligned*)

This constructor creates a PER input stream object that references an encoded ASN.1 message. In this case, the message is passed in using an System.IO.Stream object.

Parameters:

istream Input stream containing an encoded ASN.1 message.

aligned Boolean specifying PER aligned or unaligned encoding.

Member Function Documentation

virtual int **Available ()** [virtual]

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 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.

virtual void Close () [virtual]

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

Exceptions:

System.SystemException if an I/O error occurs.

override 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.

virtual bool MarkSupported () [virtual]

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`.

override 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.

override 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

Asn1PerMessageBuffer Interface Reference

Com::Objsys::Asn1::Runtime::Asn1PerMessageBufferInherited by [Asn1PerDecodeBuffer](#), and [Asn1PerEncodeBuffer](#).

Detailed Description

This interface defines constants and methods specific to encoding and decoding PER messages. All of the PER message buffer classes implement this interface.

Public Member Functions

- void [ByteAlign](#) ()
- System.IO.Stream [GetInputStream](#) ()
- bool [IsAligned](#) ()

Properties

- int [MsgBitCnt](#)
 - [Asn1PerTraceHandler](#) [TraceHandler](#)
-

Member Function Documentation

void [ByteAlign](#) ()

This method handles byte-alignment for aligned PER encoding or decoding.

Implemented in [Asn1PerDecodeBuffer](#), and [Asn1PerEncodeBuffer](#).System.IO.Stream [GetInputStream](#) ()

This method returns an input stream object that represents the message being encoded or decoded.

Returns:

Stream containing message

Implemented in [Asn1PerEncodeBuffer](#).bool [IsAligned](#) ()

This method returns a flag indicating if PER aligned encoding is currently enabled (if false, unaligned is in effect).

Returns:

true is aligned; otherwise false

Implemented in [Asn1PerDecodeBuffer](#), and [Asn1PerEncodeBuffer](#).

Property Documentation

int [MsgBitCnt](#) [get]

Gets the number of bits in the PER message.

Value: Count of bits in message

Implemented in [Asn1PerDecodeBuffer](#), and [Asn1PerEncodeBuffer](#).[Asn1PerTraceHandler](#) [TraceHandler](#) [get]

Gets a reference to the internal trace handler object used to trace the bit fields within a PER message.

Value: [Asn1PerTraceHandler](#) object

Implemented in [Asn1PerDecodeBuffer](#), and [Asn1PerEncodeBuffer](#).

Asn1PerOutputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerOutputStream

Detailed Description

This class handles the output stream for encoding of ASN.1 messages as specified in the Packed Encoding Rules (PER) ITU-T X.691 standard.

Public Member Functions

- virtual void [AddCaptureBuffer](#) (System.IO.MemoryStream buffer)
- [Asn1PerOutputStream](#) (System.IO.Stream os, int bufSize, bool aligned)
- [Asn1PerOutputStream](#) (System.IO.Stream os, bool aligned)
- virtual void [BinDump](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [BinDump](#) (System.String varName)
- virtual void [ByteAlign](#) ()
- override void [Close](#) ()
- virtual void [EncodeBit](#) (bool value, System.String ident)
- virtual void [EncodeBit](#) (bool value)
- virtual void [EncodeBits](#) (byte[] value, int offset, int nbits, System.String ident)
- virtual void [EncodeBits](#) (byte[] value, int offset, int nbits)
- virtual void [EncodeBits](#) (byte value, int nbits)
- virtual void [EncodeCharString](#) (System.String value, int nchars, int offset, int abpc, int ubpc, Asn1CharSet charSet)
- virtual void [EncodeConsWholeNumber](#) (long adjustedValue, long rangeValue)
- virtual void [EncodeConsWholeNumber](#) (long adjustedValue, long rangeValue, System.String ident)
- virtual void [EncodeInt](#) (long value, bool encodeLen, bool signExtend)
- virtual void [EncodeInt](#) (long value, bool encodeLen, bool signExtend, System.String ident)
- virtual void [EncodeInt](#) (long value, int nbits)
- virtual void [EncodeInt](#) (long value, int nbits, System.String ident)
- virtual void [EncodeLength](#) (long value, long lower, long upper)
- virtual long [EncodeLength](#) (long value)
- virtual void [EncodeLengthEOM](#) (long value)
- virtual void [EncodeOctetString](#) (byte[] value, int offset, int nbytes)
- virtual void [EncodeOIDLengthAndValue](#) (int[] value)
- virtual void [EncodeOpenType](#) (byte[] value, int offset, int nbytes)
- virtual void [EncodeRelOIDLengthAndValue](#) (int[] value)
- virtual void [EncodeSmallNonNegWholeNumber](#) (int value)
- override void [Flush](#) ()
- virtual void [RemoveCaptureBuffer](#) (System.IO.MemoryStream buffer)
- override void [Write](#) (System.Byte[] b, int off, int len)
- override void [Write](#) (byte[] b)
- override void [WriteByte](#) (byte b)
- override void [WriteByte](#) (int b)

Protected Attributes

- internal [Asn1PerOutputStreamTraceHandler mTraceHandler](#)

Properties

- virtual bool [Aligned](#)
 - virtual [Asn1PerTraceHandler TraceHandler](#)
-

Constructor & Destructor Documentation

[Asn1PerOutputStream](#) (**System.IO.Stream** *os*, **bool** *aligned*)

This constructor creates a buffered PER output stream object with the default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.
aligned Indicates whether PER aligned or unaligned encoding should be done.

[Asn1PerOutputStream](#) (**System.IO.Stream** *os*, **int** *bufSize*, **bool** *aligned*)

This constructor creates a buffered PER output stream object with the specified size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.
bufSize The buffer size.
aligned `true` for PER aligned or `false` for PER unaligned encoding.

Member Function Documentation

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

virtual void BinDump (**System.IO.StreamWriter** *outs*, **System.String** *varName*) [**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 StreamWriter object to which output should be written
varName Name of top-level message object variable

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 top-level message object variable

virtual void ByteAlign () [**virtual**]

This methods byte-aligns the buffer.

override void Close ()

Close the stream. Writes all bytes (even unfinished) before closing. Throws, exception thrown by the underlying System.IO.Stream object.

virtual void EncodeBit (bool value, System.String ident) [virtual]

This method encodes a single bit value.

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

Parameters:

value Boolean value of bit to be encoded.

ident Bit field identifier name for tracing.

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeBit (bool value) [virtual]

This method encodes a single bit value.

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

Parameters:

value Boolean value of bit to be encoded.

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeBits (byte[] value, int offset, int nbits, System.String ident) [virtual]

This method encodes bit values from an array of octets.

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

Parameters:

value Octet array containing bits to be encoded

offset Starting byte offset in value

nbits Number of bits to encode

ident Bit field identifier name for tracing.

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeBits (byte[] value, int offset, int nbits) [virtual]

This method encodes bit values from an array of octets.

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

Parameters:

value Octet array containing bits to be encoded

offset Starting byte offset in value

nbits Number of bits to encode

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeBits (byte value, int nbits) [virtual]

This method encodes bit values from an octet. The most significant bits from the octet are encoded.

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

Parameters:

value Octet containing bits to be encoded

nbits Number of bits to encode

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeCharString (System.String value, int nchars, int offset, int abpc, int ubpc, Asn1CharSet charSet) [virtual]

This method encodes the contents of a known-multiplier character string type. This version assumes a permitted alphabet constraint was specified.

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

Parameters:

value String containing characters to encode
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 Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeConsWholeNumber (long adjustedValue, long rangeValue) [virtual]

This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.

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

Parameters:

adjustedValue Adjusted value to be encoded = value - lower range endpoint value
rangeValue lower - upper + 1

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeConsWholeNumber (long adjustedValue, long rangeValue, System.String ident) [virtual]

This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.

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

Parameters:

adjustedValue Adjusted value to be encoded = value - lower range endpoint value
rangeValue lower - upper + 1
ident Bit field identifier name for tracing.

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeInt (long value, bool encodeLen, bool signExtend) [virtual]

This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.

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

Parameters:

value Integer value to be encoded
encodeLen Flag indicating length determinant should be encoded before encoding integer value.
signExtend Flag indicating if sign extension should be performed.

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeInt (long value, bool encodeLen, bool signExtend, System.String ident) [virtual]

This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.

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

Parameters:

value Integer value to be encoded
encodeLen Flag indicating length determinant should be encoded before encoding integer value.
signExtend Flag indicating if sign extension should be performed.
ident Bit field identifier name for tracing.

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeInt (long value, int nbits) [virtual]

This method encodes bit values from an integer value. The least significant bits from the integer are encoded.

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

Parameters:

value Integer containing bits to be encoded
nbits Number of bits to encode

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeInt (long value, int nbits, System.String ident) [virtual]

This method encodes bit values from an integer value. The least significant bits from the integer are encoded.

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

Parameters:

value Integer containing bits to be encoded
nbits Number of bits to encode
ident Bit field identifier name for tracing.

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeLength (long value, long lower, long upper) [virtual]

This method encodes a constrained length determinant value.

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

Parameters:

value Length value to be encoded
lower Lower bound (inclusive) of length value range
upper Upper bound (inclusive) of length value range

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual long EncodeLength (long value) [virtual]

This method encodes a general (unconstrained) length determinant value as described in section 10.9 or the X.691 standard.

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

Parameters:

value Length value to be encoded

Returns:

Value that was actually encoded. This may be less than the value that was passed in if fragmentation was done (i.e the value was $\geq 16k$).

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeLengthEOM (long value) [virtual]

This method checks to see if a zero byte needs to be added after a fragmented length has been encoded. It will add it to the byte stream if necessary.

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

Parameters:

value Original length value that was encoded.

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeOctetString (byte[] value, int offset, int nbytes) [virtual]

This method encodes the given array of bytes as an unconstrained octet string value.

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

Parameters:

value Byte array containing data to encode. This is assumed to contain a previously encoded PER component.
offset Starting offset in byte array value
nbytes Number of bytes to encode

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeOIDLengthAndValue (int[] value) [virtual]

This method encodes the length and contents of an object identifier value.

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

Parameters:

value Integer array containing arcs to encode

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeOpenType (byte[] value, int offset, int nbytes) [virtual]

This method encodes the given array of bytes as an open type.

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

Parameters:

value Byte array containing data to encode. This is assumed to contain a previously encoded PER component.

offset Starting offset in byte array value

nbytes Number of bytes to encode

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeRelOIDLengthAndValue (int[] value) [virtual]

This method encodes the length and contents of a relative object identifier value.

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

Parameters:

value Integer array containing arcs to encode

Exceptions:

Asn1Exception Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

virtual void EncodeSmallNonNegWholeNumber (int value) [virtual]

This method implements the rules to encode a small non-negative whole number as specified in section 10.6 of the X.691 standard.

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

Parameters:

value Value to be encoded

Exceptions:

Asn1InvalidArgException Any exception thrown by the underlying [Asn1PerEncodeBuffer](#).

override void Flush ()

Flush the buffer to the stream. It writes whole bytes only. Throws, exception thrown by the underlying System.IO.Stream object.

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

override void Write (System.Byte[] b, int off, int len)

Writes *len* bytes from the specified byte array to this output stream.

Parameters:

b the data.
off The offset in array at which to begin write.
len The number of bytes to write.

Exceptions:

System.SystemException if an I/O error occurs. In particular, write call after the the output stream is closed.

override void Write (byte[] b)

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 binary data.

Exceptions:

System.SystemException if an I/O error occurs.

override void WriteByte (byte b)

Writes the specified byte to this output stream.

Parameters:

b the byte.

Exceptions:

System.SystemException if an I/O error occurs. In particular, an write call after the output stream has been closed.

override void WriteByte (int b)

Writes the specified byte to this output stream.

Parameters:

b the byte.

Exceptions:

System.SystemException if an I/O error occurs. In particular, an write call after the output stream has been closed.

Member Data Documentation

internal [Asn1PerOutputStreamTraceHandler](#) [mTraceHandler](#) [protected]

Variable holds the PER message trace handler

Property Documentation

virtual bool **Aligned** [get]

Gets PER aligned encoding has been specified.

Value: true is aligned; otherwise false

virtual [Asn1PerTraceHandler](#) TraceHandler [get]

Gets a reference to the internal trace handler object used to trace the bit fields within a PER message.

Value: [Asn1PerTraceHandler](#) object

Asn1PerOutputStreamTraceHandler Class Reference

Com::ObjSys::Asn1::Runtime::Asn1PerOutputStreamTraceHandlerInherits [Asn1PerTraceHandler](#).

Detailed Description

This is a utility class for handling the collection and printing of PER bit field trace information for PER output stream. An object of the class is present in the [Asn1PerOutputStream](#) classes. It is accessed using the 'TraceHandler' property from objects of these classes.

Public Member Functions

- [Asn1PerOutputStreamTraceHandler](#) ([Asn1PerOutputStream](#) outs)
 - override void [Enable](#) ()
 - override void [Print](#) (System.IO.StreamWriter outs, System.String varName)
 - override void [Reset](#) ()
 - virtual void [ResetTrace](#) ()
-

Constructor & Destructor Documentation

[Asn1PerOutputStreamTraceHandler](#) ([Asn1PerOutputStream](#) outs)

This constructor initializes the internal trace handler member variables.

Parameters:

outs PER message stream object reference

Member Function Documentation

override void [Enable](#) () [virtual]

This method is used to turn PER bit tracing on

Implements [Asn1PerTraceHandler](#).override void [Print](#) (System.IO.StreamWriter *outs*, System.String *varName*) [virtual]

This method prints the trace to the given output stream in a default format.

Parameters:

- outs* Print stream to which output is to be written.
- varName* Name of the object variable being printed.

Implements [Asn1PerTraceHandler](#).override void Reset () [virtual]

This method does nothing here.

Implements [Asn1PerTraceHandler](#).virtual void ResetTrace () [virtual]

This method resets the trace bit field list.

Asn1PerTraceHandler Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerTraceHandlerInherited by [Asn1PerDecodeTraceHandler](#), [Asn1PerEncodeTraceHandler](#), and [Asn1PerOutputStreamTraceHandler](#).

Detailed Description

This is the abstract base class for the PER encode and decode trace handler derived classes.

Public Member Functions

- virtual void [AddElemName](#) (System.String name, int arrayx)
- abstract void [Enable](#) ()
- virtual void [NewBitField](#) (System.String name, int bitCount)
- abstract void [Print](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [RemoveLastElemName](#) ()
- abstract void [Reset](#) ()
- virtual void [SetBitCount](#) ()
- virtual void [SetBitOffset](#) ()

Protected Member Functions

- internal [Asn1PerTraceHandler](#) ([Asn1PerMessageBuffer](#) messageBuffer)

Protected Attributes

- internal [Asn1PerBitFieldList](#) [mBitFieldList](#)

Properties

- virtual [Asn1PerBitFieldList](#) [BitFieldList](#)

Constructor & Destructor Documentation

internal [Asn1PerTraceHandler](#) ([Asn1PerMessageBuffer](#) messageBuffer) [protected]

This constructor initializes internal trace handler member variables.

Parameters:

messageBuffer PER message buffer object reference

Member Function Documentation**virtual void AddElemName (System.String *name*, int *arrayx*) [virtual]**

This method adds an element name to the current fully qualified name. The fully qualified name is a string of name components separated by dots (ex. a.b.c).

Parameters:

name Name component to append to string

arrayx Array index if named item is an element in an array (set to -1 otherwise)

abstract void Enable () [pure virtual]

This method is used to turn PER bit tracing on or off

Implemented in [Asn1PerEncodeTraceHandler](#), [Asn1PerDecodeTraceHandler](#), and [Asn1PerOutputStreamTraceHandler](#). **virtual void NewBitField (System.String *name*, int *bitCount*) [virtual]**

This method creates a new bit field and appends it to the bit field list.

Parameters:

name Name suffix to append to the current fully qualified name.

bitCount Number of bits in the bit field.

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

This method prints the trace to the given output stream in a default format.

Parameters:

outs Print stream to which output is to be written.

varName Name of the object variable being printed.

Implemented in [Asn1PerEncodeTraceHandler](#), [Asn1PerDecodeTraceHandler](#), and [Asn1PerOutputStreamTraceHandler](#). **virtual void RemoveLastElemName () [virtual]**

This method removes the last element name in the current fully qualified name string. For example, if the current string is 'a.b.c', it will be 'a.b' after calling this method.

abstract void Reset () [pure virtual]

This method resets the trace bit field list.

Implemented in [Asn1PerEncodeTraceHandler](#), [Asn1PerDecodeTraceHandler](#), and [Asn1PerOutputStreamTraceHandler](#). **virtual void SetBitCount () [virtual]**

This method sets the bit count within the current bit field to the difference between the current bit offset and the starting bit offset currently stored in the field object.

virtual void SetBitOffset () [virtual]

This method sets the bit offset within the current bit field to the current offset within the PER message buffer.

Member Data Documentation

internal [Asn1PerBitFieldList](#) [mBitFieldList](#) [protected]

Variable holds the bit field list

Property Documentation

virtual [Asn1PerBitFieldList](#) [BitFieldList](#) [get]

Gets a reference to the bit field list

Value: bit field list

Asn1PerUtil Class Reference

Com::Objsys::Asn1::Runtime::Asn1PerUtil

Detailed Description

This class contains general purpose static utility functions related to PER encoding/decoding

Static Public Member Functions

- static int [GetMsgBitCnt](#) (int *byteCount*, int *bitOffset*)
-

Member Function Documentation

static int [GetMsgBitCnt](#) (int *byteCount*, int *bitOffset*) [static]

This method returns the number of bits in an encoded PER message buffer.

Parameters:

byteCount Number of full bytes in message

bitOffset Offset to current bit in last byte ((7 - 0) or -1 if no bits used).

Returns:

Count of bits in encoded message

Asn1SetDuplicateException Class Reference

Com::Objsys::Asn1::Runtime::Asn1SetDuplicateException

Detailed Description

This class defines the 'ASN.1 set duplicate element' exception that is thrown from BER/DER methods when a SET construct is detected to more than one instance of a given tagged element..

Public Member Functions

- [Asn1SetDuplicateException](#) ([Asn1BerDecodeBuffer](#) buffer, Asn1Tag tag)
-

Constructor & Destructor Documentation

[Asn1SetDuplicateException](#) ([Asn1BerDecodeBuffer](#) *buffer*, Asn1Tag *tag*)

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

Parameters:

buffer BER decode buffer object reference
tag Tag value of duplicate element

Asn1TaggedEventHandler Interface Reference

Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandlerInherited
[Asn1BerMessageDumpHandler](#).

by

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.

A tagged event handler differs from a named event handler in

that it returns the tags from within a BER or DER message instead of the symbolic names. This type of handler can be used to generically parse a message without knowledge of the associated ASN.1 schema definition. It is used in conjunction with the [Asn1BerDecodeBuffer](#) *Parse* method.

Public Member Functions

- void [Contents](#) (byte[] data)
 - void [EndElement](#) (Asn1Tag tag)
 - void [StartElement](#) (Asn1Tag tag, int len, byte[] tagLenBytes)
-

Member Function Documentation

void Contents (byte[] data)

The contents callback method is invoked when the contents of a primitive data element are parsed.

Parameters:

data Array containing encoded contents bytes.

Implemented in [Asn1BerMessageDumpHandler](#).void EndElement (Asn1Tag tag)

The endElement callback method is invoked when the end of a tagged element is parsed.

Parameters:

tag Parsed tag value.

Implemented in [Asn1BerMessageDumpHandler](#).void StartElement (Asn1Tag tag, int len, byte[] tagLenBytes)

The StartElement callback method is invoked when the start of any tagged element is parsed.

Parameters:

tag Parsed tag value.

len Parsed length value

tagLenBytes Array containing the encoded bytes that make up the tag/length sequence.

Implemented in [Asn1BerMessageDumpHandler](#).

Asn1TagMatchFailedException Class Reference

Com::Objsys::Asn1::Runtime::Asn1TagMatchFailedException

Detailed Description

This class defines the 'ASN.1 tag match failed' exception that is thrown from BER/DER methods when an expected tag is not matched..

Public Member Functions

- [Asn1TagMatchFailedException](#) ([Asn1BerDecodeBuffer](#) buffer, Asn1Tag expectedTag)
- [Asn1TagMatchFailedException](#) ([Asn1BerDecodeBuffer](#) buffer, Asn1Tag expectedTag, Asn1Tag parsedTag)

Constructor & Destructor Documentation

[Asn1TagMatchFailedException](#) ([Asn1BerDecodeBuffer](#) buffer, Asn1Tag expectedTag, Asn1Tag parsedTag)

This constructor creates an exception object with a textual message describing the expected and parsed tag values..

Parameters:

buffer BER decode buffer object reference
expectedTag Expected tag value
parsedTag Parsed tag value

[Asn1TagMatchFailedException](#) ([Asn1BerDecodeBuffer](#) *buffer*, [Asn1Tag](#) *expectedTag*)

This constructor creates an exception object with a textual message describing only the expected tag value. It is used in cases where the parsed tag value cannot be determined.

Parameters:

buffer BER decode buffer object reference
expectedTag Expected tag value

Asn1XerBase64OctetString Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerBase64OctetString

Detailed Description

This is a container class for holding the components of an ASN.1 octet string value. This is a special version of the class that is only generated for the XER encoding rules, if Base64 encoding is used.

Public Member Functions

- [Asn1XerBase64OctetString](#) (System.String value)
- [Asn1XerBase64OctetString](#) (byte[] data, int offset, int nbytes)
- [Asn1XerBase64OctetString](#) (byte[] data)
- [Asn1XerBase64OctetString](#) ()
- override void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer, System.String elemName, System.String attribute)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer, System.String elemName)

Constructor & Destructor Documentation

[Asn1XerBase64OctetString](#) ()

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

[Asn1XerBase64OctetString](#) (byte[] *data*)

This constructor initializes an octet string from the given byte array.

Parameters:

data Byte array containing an octet string in binary form.

Asn1XerBase64OctetString (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 Starting offset in data from which to copy bytes
nbytes Number of bytes to copy from target array

Asn1XerBase64OctetString (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

Member Function Documentation

override void DecodeXER (System.String buffer, System.String attrs)

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

Parameters:

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

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

override void Encode ([Asn1XerEncoder](#) buffer, System.String elemName, System.String attribute)

This method encodes ASN.1 octet string type with element and attribute name tag according to 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

override void Encode ([Asn1XerEncoder](#) buffer, System.String elemName)

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

Asn1XerDecodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer

Detailed Description

This class handles the decoding of ASN.1 messages as specified in the XML Encoding Rules (XER) as documented in the ITU-T X.693 standard. Note that this class is not derived from the `Asn1DecodeBuffer` class as are other decode buffer classes. Its purpose is to act as an input source for XML data to be read by a SAX parser.

Public Member Functions

- [Asn1XerDecodeBuffer](#) (System.String source)

Protected Attributes

- internal [XmlSource](#) [mInputSource](#)

Properties

- virtual [XmlSource](#) [InputSource](#)
-

Constructor & Destructor Documentation

[Asn1XerDecodeBuffer](#) (System.String source)

This constructor creates an XER decode buffer.

Parameters:

source The source containing the XML document.

Member Data Documentation

internal [XmlSource](#) [mInputSource](#) [protected]

Variable holds the SAX input source object.

Property Documentation

virtual [XmlSource](#) [InputSource](#) [get]

Gets the SAX input source object.

Value: SAX input source object

Asn1XerElemInfo Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerElemInfo

Detailed Description

This class holds XER element information needed to assign an identifier to an element after it is parsed from an XML message.

Public Member Functions

- [Asn1XerElemInfo](#) (System.String name, bool optional)
- bool [Equals](#) (System.String name)
- override int [GetHashCode](#) ()

Properties

- virtual bool [Optional](#)
-

Constructor & Destructor Documentation

[Asn1XerElemInfo](#) (System.String name, bool optional)

This constructor creates the element object

Parameters:

name element name
optional true if element is optional

Member Function Documentation

bool Equals (System.String name)

Determines whether the specified Object is equal to the given String.

Parameters:

name The String to compare with the current Object.

Returns:

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

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.

Property Documentation

virtual bool Optional [get]

Determines whether the this element is optional

Value: true is optional; otherwise false

Asn1XerEncodeBuffer Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerEncodeBufferInherits [Asn1XerEncoder](#).

Inherited by [Asn1XmlEncodeBuffer](#).

Detailed Description

This class handles the encoding of ASN.1 messages as specified in the XML Encoding Rules (XER) as specified in the ITU-T X.693 standard.

Public Member Functions

- [Asn1XerEncodeBuffer](#) (bool canonical, int sizeIncrement)
- [Asn1XerEncodeBuffer](#) (bool canonical)
- [Asn1XerEncodeBuffer](#) ()
- override void [BinDump](#) (System.IO.StreamWriter outs, System.String varName)
- virtual void [Copy](#) (System.String data)
- virtual void [Copy](#) (byte[] data, int off, int len)
- override void [Copy](#) (byte[] data)
- override void [Copy](#) (byte data)
- virtual void [DecrLevel](#) ()
- virtual void [EncodeBinStrValue](#) (byte[] bits, int nbits)
- virtual void [EncodeByte](#) (byte data)
- virtual void [EncodeData](#) (System.String data)
- virtual void [EncodeEmptyElement](#) (System.String elemName, System.String attribute)
- virtual void [EncodeEmptyElement](#) (System.String elemName)
- virtual void [EncodeEndDocument](#) ()
- virtual void [EncodeEndElement](#) (System.String elemName)
- virtual void [EncodeHexStrValue](#) (byte[] data)
- virtual void [EncodeNamedValue](#) (System.String valueName, System.String elemName, System.String attribute)
- virtual void [EncodeNamedValue](#) (System.String valueName, System.String elemName)
- virtual void [EncodeNamedValueElement](#) (System.String elemName)
- virtual void [EncodeObjectId](#) (int[] data)
- virtual void [EncodeRealValue](#) (double data, System.String elemName, System.String attribute)
- virtual void [EncodeRealValue](#) (double data, System.String elemName)
- virtual void [EncodeStartDocument](#) ()
- virtual void [EncodeStartElement](#) (System.String elemName, System.String attribute)
- virtual void [EncodeStartElement](#) (System.String elemName)

- override System.IO.Stream [GetInputStream](#) ()
- virtual void [IncrLevel](#) ()
- virtual void [Indent](#) ()
- override void [Reset](#) ()
- override void [Write](#) (System.IO.Stream outs)

Protected Member Functions

- internal override void [CheckSize](#) (int bytesRequired)

Properties

- virtual byte[] [Buffer](#)
- virtual bool [Canonical](#)
- override byte[] [MsgCopy](#)
- override int [MsgLength](#)
- virtual int [State](#)

Constructor & Destructor Documentation

[Asn1XerEncodeBuffer](#) ()

The default constructor creates an XER encode buffer object with the default size increment and canonical set to false.

[Asn1XerEncodeBuffer](#) (bool *canonical*)

The parameterized constructor creates an XER encode buffer object with default size increment and canonical set to the given values.

Parameters:

canonical Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.

[Asn1XerEncodeBuffer](#) (bool *canonical*, int *sizeIncrement*)

The parameterized constructor creates an XER encode buffer object with size increment and canonical set to the given values.

Parameters:

canonical Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.

sizeIncrement The initial size in bytes of an encode buffer. If the buffer becomes full, it will be expanded by the amount. If this parameter is set to zero, the default increment will be used.

Member Function Documentation

override void [BinDump](#) (System.IO.StreamWriter *outs*, System.String *varName*)

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

internal override void CheckSize (int *bytesRequired*) [protected]

This method determines if the encode buffer can hold the requested number of bytes. If not, the buffer is expanded.

Parameters:

bytesRequired Number of required bytes.

virtual void Copy (System.String *data*) [virtual]

This method copies a character string to the encode buffer.

Parameters:

data The string value to copy

Implements [Asn1XerEncoder](#).virtual void Copy (byte[] *data*, int *off*, int *len*) [virtual]

This method copies multiple bytes to the encode buffer. It is assumed the byte are already formatted into a valid XML encoding type (for example, UTF-8).

Parameters:

data Array of bytes to copy to the encode buffer

off The offset in array at which to begin copy.

len The number of bytes to copy

Implements [Asn1XerEncoder](#).override void Copy (byte[] *data*)

This method copies multiple bytes to the encode buffer. It is assumed the byte are already formatted into a valid XML encoding type (for example, UTF-8).

Parameters:

data Array of bytes to copy to the encode buffer

Implements [Asn1XerEncoder](#).override void Copy (byte *data*)

This method is used to copy a single byte to the encode buffer. It first converts the byte to a hex character representation and then copies it to the output buffer.

Parameters:

data The byte value to copy

Implements [Asn1XerEncoder](#).virtual void DecrLevel () [virtual]

This method decrements the element nesting level counter.

Implements [Asn1XerEncoder](#).virtual void EncodeBinStrValue (byte[] *bits*, int *nbits*) [virtual]

This method encodes XML binary string data

Parameters:

bits Bit String to encode

nbits Number of bits to encode

Implements [Asn1XerEncoder](#).virtual void EncodeByte (byte *data*) [virtual]

This method is used to encode a single byte to the encode buffer. It first converts the byte to a hex character representation and then copies it to the output buffer.

Parameters:

data The byte value to copy

Implements [Asn1XerEncoder](#).virtual void EncodeData (System.String *data*) [virtual]

This method encodes XML string data

Parameters:

data String value to encode

Implements [Asn1XerEncoder](#).virtual void EncodeEmptyElement (System.String *elemName*, System.String *attribute*) [virtual]

This method encodes an XML empty element tag

Parameters:

elemName The name of element.

attribute The name and value of attribute.

Implements [Asn1XerEncoder](#).virtual void EncodeEmptyElement (System.String *elemName*) [virtual]

This method encodes an XML empty element tag

Parameters:

elemName The name of element.

Implements [Asn1XerEncoder](#).virtual void EncodeEndDocument () [virtual]

This method encodes standard trailer information at the end of the XML document.

Implements [Asn1XerEncoder](#).virtual void EncodeEndElement (System.String *elemName*) [virtual]

This method encodes an XML end element tag

Parameters:

elemName The name of element.

Implements [Asn1XerEncoder](#).Reimplemented in [Asn1XmlEncodeBuffer](#).virtual void EncodeHexStrValue (byte[] *data*) [virtual]

This method encodes XML hexadecimal string data

Parameters:

data Data to encode

Implements [Asn1XerEncoder](#).virtual void EncodeNamedValue (System.String *valueName*, System.String *elemName*, System.String *attribute*) [virtual]

This method encodes an XML named value (with start and end tags)

Parameters:

valueName The named value.

elemName The name of element.

attribute The name and value of attribute.

Implements [Asn1XerEncoder](#).virtual void EncodeNamedValue (System.String *valueName*, System.String *elemName*) [virtual]

This method encodes an XML named value (with start and end tags)

Parameters:

elemName The name of element.
valueName named value.

Implements [Asn1XerEncoder](#).virtual void EncodeNamedValueElement (System.String *elemName*) [virtual]

This method encodes an XML named value element tag

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).Reimplemented in [Asn1XmlEncodeBuffer](#).virtual void EncodeObjectIds (int[] *data*) [virtual]

This method encodes XML Object Identifiers and Relative OIDs data

Parameters:

data Object's identifiers to encode

Implements [Asn1XerEncoder](#).virtual void EncodeRealValue (double *data*, System.String *elemName*, System.String *attribute*) [virtual]

This method encodes an XML REAL (double) value (with start and end tags).

Parameters:

data The value to be encoded.
elemName The name of element. If null, then start and end tags won't be encoded.
attribute The name and value of attribute.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeRealValue (double *data*, System.String *elemName*) [virtual]

This method encodes an XML REAL (double) value (with start and end tags).

Parameters:

data The value to be encoded.
elemName The name of element. If null, then start and end tags won't be encoded.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeStartDocument () [virtual]

This method encodes standard header information at the beginning of the XML document.

Implements [Asn1XerEncoder](#).virtual void EncodeStartElement (System.String *elemName*, System.String *attribute*) [virtual]

This method encodes an XML start element tag with attribute

Parameters:

elemName The name of element.

attribute The name and value of attribute.

Implements [Asn1XerEncoder](#).virtual void EncodeStartElement (System.String elemName) [virtual]

This method encodes an XML start element tag

Parameters:

elemName The name of element.

Implements [Asn1XerEncoder](#).override System.IO.Stream GetInputStream ()

This method returns an input stream object reference to the message buffer contents (i.e. the encoded data). If an output stream was selected as the output method, this method returns null.

Returns:

Input stream object reference

virtual void IncrLevel () [virtual]

This method increments the element nesting level counter.

Implements [Asn1XerEncoder](#).virtual void Indent () [virtual]

This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the encode buffer.

Implements [Asn1XerEncoder](#).override void Reset ()

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

override void Write (System.IO.Stream outs)

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

Parameters:

outs Output stream to which record is to be written

Property Documentation

virtual byte [] Buffer [get]

Gets a reference of the byte buffer used to hold the encoded message.

Value: byte array containing encoded message

virtual bool Canonical [set]

Sets the canonical encoding rule.

Value: true if canonical encoding; otherwise false .

override byte [] MsgCopy [get]

Gets the copy of the byte buffer used to hold the encoded message.

Value: byte array containing encoded message

override int MsgLength [get]

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

Value: length of encoded message component

virtual int State [set]

Sets the state of the buffer.

Value: buffer stat

Implements [Asn1XerEncoder](#).

Asn1XerEncoder Interface Reference

Com::Objsys::Asn1::Runtime::Asn1XerEncoderInherited by [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).

Detailed Description

This is a base interface for encoding of ASN.1 messages as specified in the XML Encoding Rules (XER) as specified in the ITU-T X.693 standard. It is implemented by both the [Asn1XerEncodeBuffer](#) and [Asn1XerOutputStream](#).

Public Member Functions

- void [Copy](#) (System.String data)
- void [Copy](#) (byte[] data, int off, int len)
- void [Copy](#) (byte[] data)
- void [Copy](#) (byte data)
- void [DecrLevel](#) ()
- void [EncodeBinStrValue](#) (byte[] bits, int nbits)
- void [EncodeByte](#) (byte data)
- void [EncodeData](#) (System.String data)
- void [EncodeEmptyElement](#) (System.String elemName, System.String attribute)
- void [EncodeEmptyElement](#) (System.String elemName)
- void [EncodeEndDocument](#) ()
- void [EncodeEndElement](#) (System.String elemName)
- void [EncodeHexStrValue](#) (byte[] data)
- void [EncodeNamedValue](#) (System.String valueName, System.String elemName, System.String attribute)
- void [EncodeNamedValue](#) (System.String valueName, System.String elemName)
- void [EncodeNamedValueElement](#) (System.String elemName)
- void [EncodeObjectId](#) (int[] data)

- void [EncodeRealValue](#) (double valueName, System.String elemName, System.String attribute)
- void [EncodeRealValue](#) (double valueName, System.String elemName)
- void [EncodeStartDocument](#) ()
- void [EncodeStartElement](#) (System.String elemName, System.String attribute)
- void [EncodeStartElement](#) (System.String elemName)
- void [IncrLevel](#) ()
- void [Indent](#) ()

Properties

- int [State](#)

Member Function Documentation

void Copy (System.String data)

This method copies a character string to the encode buffer or stream.

Throws C# Exception, If I/O error occurs.

Parameters:

data The string value to copy

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void Copy (byte[] data, int off, int len)

This method copies multiple bytes to the encode buffer or stream. It is assumed the byte are already formatted into a valid XML encoding type (for example, UTF-8).

Throws C# Exception, If I/O error occurs.

Parameters:

data Array of bytes to copy to the encode buffer

off The offset in array at which to begin copy.

len The number of bytes to copy

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void Copy (byte[] data)

This method copies multiple bytes to the encode buffer or stream. It is assumed the byte are already formatted into a valid XML encoding type (for example, UTF-8).

Throws C# Exception, If I/O error occurs.

Parameters:

data Array of bytes to copy to the encode buffer

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void Copy (byte data)

This method is used to copy a single byte to the encode buffer or stream. It first converts the byte to a hex character representation and then copies it to the output buffer.

Throws C# Exception, If I/O error occurs.

Parameters:

data The byte value to copy

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void DecrLevel ()

This method decrements the element nesting level counter.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeBinStrValue (byte[] *bits*, int *nbits*)

This method encodes XML binary string data

Throws C# Exception, If I/O error occurs.

Parameters:

bits Bit String to encode

nbits Number of bits to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeByte (byte *data*)

This method is used to encode a single byte to the encode buffer or stream. It first converts the byte to a hex character representation and then copies it to the output buffer.

Parameters:

data The byte value to copy

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeData (System.String *data*)

This method encodes XML string data

Throws C# Exception, If I/O error occurs.

Parameters:

data String value to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeEmptyElement (System.String *elemName*, System.String *attribute*)

This method encodes an XML empty element tag. element name tag will also contain the attribute name and value

Throws C# Exception, If I/O error occurs.

Parameters:

elemName The name of element.

attribute The name and value of attribute.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeEmptyElement (System.String elemName)

This method encodes an XML empty element tag.

Throws C# Exception, If I/O error occurs.

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeEndDocument ()

This method encodes standard trailer information at the end of the XML document.

Throws C# Exception, If I/O error occurs.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeEndElement (System.String elemName)

This method encodes an XML end element tag.

Throws C# Exception, If I/O error occurs.

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), [Asn1XmlEncodeBuffer](#), [Asn1XerOutputStream](#), and [Asn1XmlOutputStream](#).void EncodeHexStrValue (byte[] data)

This method encodes XML hexadecimal string data

Throws C# Exception, If I/O error occurs.

Parameters:

data Data to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeNamedValue (System.String valueName, System.String elemName, System.String attribute)

This method encodes an XML named value (with start and end tags). start tag will contain the attribute name and value

Throws C# Exception, If I/O error occurs.

Parameters:

valueName The named value.

elemName The name of element.

attribute The name and value of attribute.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void **EncodeNamedValue (System.String valueName, System.String elemName)**

This method encodes an XML named value (with start and end tags).

Throws C# Exception, If I/O error occurs.

Parameters:

valueName The name of value.

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void **EncodeNamedValueElement (System.String elemName)**

This method encodes an XML named value element tag.

Throws C# Exception, If I/O error occurs.

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), [Asn1XmlEncodeBuffer](#), [Asn1XerOutputStream](#), and [Asn1XmlOutputStream](#).void **EncodeObjectId (int[] data)**

This method encodes XML Object Identifiers and Relative OIDs data

Throws C# Exception, If I/O error occurs.

Parameters:

data Object's identifiers to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void **EncodeRealValue (double valueName, System.String elemName, System.String attribute)**

This method encodes an XML REAL (double) value (with start and end tags). start tag will contain the attribute name and value

Throws C# Exception, If I/O error occurs.

Parameters:

valueName The name of value.

elemName The name of element. If null, then start and end

attribute The name and value of attribute. tags won't be encoded.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void **EncodeRealValue (double valueName, System.String elemName)**

This method encodes an XML REAL (double) value (with start and end tags).

Throws C# Exception, If I/O error occurs.

Parameters:

valueName The name of value.

elemName The name of element. If null, then start and end tags won't be encoded.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeStartDocument ()

This method encodes standard header information at the beginning of the XML document.

Throws C# Exception, If I/O error occurs.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeStartElement (System.String *elemName*, System.String *attribute*)

This method encodes an XML start element and attribute tag. start tag will contain the attribute name and value

Throws C# Exception, If I/O error occurs.

Parameters:

elemName The name of element.

attribute The name and value of attribute.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void EncodeStartElement (System.String *elemName*)

This method encodes an XML start element tag.

Throws C# Exception, If I/O error occurs.

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void IncrLevel ()

This method increments the element nesting level counter.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).void Indent ()

This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the encode buffer.

Throws C# Exception, If I/O error occurs.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).

Property Documentation

int State [set]

Sets the state of the buffer.

Value: Buffer Stat

Implemented in [Asn1XerEncodeBuffer](#), and [Asn1XerOutputStream](#).

Asn1XerEncoder_Fields Struct Reference

Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields

Detailed Description

This class defines the constant variables for [Asn1XerEncoder](#).

Static Public Attributes

- static readonly int [XERDATA](#) = 2
 - static readonly int [XEREND](#) = 3
 - static readonly int [XERINDENT](#) = 3
 - static readonly int [XERINIT](#) = 0
 - static readonly int [XERSTART](#) = 1
-

Member Data Documentation

readonly int [XERDATA](#) = 2 [static]

XER characters (data) state

readonly int [XEREND](#) = 3 [static]

XER end element state

readonly int [XERINDENT](#) = 3 [static]

Number of indent spaces required to print XER element

readonly int [XERINIT](#) = 0 [static]

XER initialization state

readonly int [XERSTART](#) = 1 [static]

XER start element state

Asn1XerOpenType Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerOpenType

Detailed Description

This is a container class for holding the an ASN.1 open type value. This is a special version of the class that is only generated for the XER encoding rules.

Public Member Functions

- [Asn1XerOpenType](#) (Asn1EncodeBuffer buffer)
- [Asn1XerOpenType](#) (byte[] data, int offset, int nbytes)
- [Asn1XerOpenType](#) (byte[] data)
- [Asn1XerOpenType](#) ()
- override void [Decode](#) ([Asn1PerDecodeBuffer](#) buffer)
- override void [Decode](#) ([Asn1BerDecodeBuffer](#) buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer, System.String elemName)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer, System.String elemName, System.String attribute)
- override void [Encode](#) ([Asn1PerEncodeBuffer](#) buffer)
- override int [Encode](#) ([Asn1BerEncodeBuffer](#) buffer, bool explicitTagging)
- virtual [Asn1XerSaxHandler](#) [GetSaxHandler](#) ()

Classes

- class [SaxHandler](#)
-

Constructor & Destructor Documentation

[Asn1XerOpenType](#) ()

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

[Asn1XerOpenType](#) (byte[] data)

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 message component.

[Asn1XerOpenType](#) (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.

Parameters:

data Byte array containing an octet string in binary form.

offset The byte offset in array at which to begin.

nbytes Number of bytes to copy from offset

[Asn1XerOpenType](#) ([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.

Parameters:

buffer Reference to encode buffer into which component type was encoded.

Member Function Documentation

override void Decode ([Asn1PerDecodeBuffer](#) *buffer*)

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

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

This method decodes an ASN.1 open type value.

Parameters:

buffer Decode message buffer object

explicitTagging Flag indicating element is explicitly tagged

implicitLength Length of contents if implicit

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

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

Parameters:

buffer Encode message buffer object

elemName Element name

override void Encode ([Asn1XerEncoder](#) *buffer*, System.String *elemName*, System.String *attribute*)

This method encodes an ASN.1 open type 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

override void Encode ([Asn1PerEncodeBuffer](#) *buffer*)

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

Parameters:

buffer Encode message buffer object

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

This method encodes an ASN.1 open type value. The value is assumed to be an already-encoded 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

virtual [Asn1XerSaxHandler](#) GetSaxHandler () [virtual]

This method returns the [Asn1XerOpenType.SaxHandler](#) class instance used for ASN.1 XER encoding.

Returns:

[Asn1XerOpenType.SaxHandler](#) object

Asn1XerOpenType.SaxHandler Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerOpenType::SaxHandlerInherits [Asn1XerSaxHandler](#).

Detailed Description

This class extends the [Asn1XerSaxHandler](#) class to add items specific to ASN.1 XER encoding.

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)

Member Function Documentation

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

override void EndElement (System.String namespaceURI, System.String localName, System.String qName) [virtual]

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

Reimplemented from [XmlSaxDefaultHandler](#). override void StartElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*, [XmlAttributes](#) *atts*) [virtual]

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

Reimplemented from [XmlSaxDefaultHandler](#).

Asn1XerOutputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerOutputStreamInherits [Asn1XerEncoder](#).

Inherited by [Asn1XmlOutputStream](#).

Detailed Description

This class implements the output stream to encode ASN.1 messages as specified in the XML Encoding Rules (XER) as specified in the ITU-T X.693 standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Public Member Functions

- [Asn1XerOutputStream](#) (System.IO.Stream os, bool canonical, int bufSize)
- [Asn1XerOutputStream](#) (System.IO.Stream os)
- virtual void [Copy](#) (System.String data)
- virtual void [Copy](#) (byte[] data, int off, int len)
- virtual void [Copy](#) (byte[] data)
- virtual void [Copy](#) (byte data)
- virtual void [DecrLevel](#) ()
- virtual void [EncodeBinStrValue](#) (byte[] bits, int nbits)
- virtual void [EncodeByte](#) (byte data)
- virtual void [EncodeData](#) (System.String data)
- virtual void [EncodeEmptyElement](#) (System.String elemName, System.String attribute)
- virtual void [EncodeEmptyElement](#) (System.String elemName)
- virtual void [EncodeEndDocument](#) ()
- virtual void [EncodeEndElement](#) (System.String elemName)
- virtual void [EncodeHexStrValue](#) (byte[] data)
- virtual void [EncodeNamedValue](#) (System.String valueName, System.String elemName, System.String attribute)

- virtual void [EncodeNamedValue](#) (System.String valueName, System.String elemName)
- virtual void [EncodeNamedValueElement](#) (System.String elemName)
- virtual void [EncodeObjectId](#) (int[] data)
- virtual void [EncodeRealValue](#) (double data, System.String elemName, System.String attribute)
- virtual void [EncodeRealValue](#) (double data, System.String elemName)
- virtual void [EncodeStartDocument](#) ()
- virtual void [EncodeStartElement](#) (System.String elemName, System.String attribute)
- virtual void [EncodeStartElement](#) (System.String elemName)
- virtual void [IncrLevel](#) ()
- virtual void [Indent](#) ()
- virtual void [Write](#) (System.String data)

Properties

- virtual bool [Canonical](#)
- virtual int [State](#)

Constructor & Destructor Documentation

[Asn1XerOutputStream](#) (System.IO.Stream os)

This constructor creates a buffered XER output stream object with default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

[Asn1XerOutputStream](#) (System.IO.Stream os, bool canonical, int bufSize)

This constructor creates a buffered XER output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

canonical Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.

bufSize The buffer size. If it is 0 then the output stream is used as unbuffered.

Member Function Documentation

virtual void Copy (System.String data) [virtual]

This method copies a character string to the output stream.

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

Parameters:

data The string value to copy

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void Copy (byte[] data, int off, int len) [virtual]

This method copies multiple bytes to the output stream. It is assumed the byte are already formatted into a valid XML encoding type (for example, UTF-8).

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

Parameters:

data Array of bytes to copy to the output stream
off The offset in array at which to begin copy.
len The Number of bytes to copy

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void Copy (byte[] data) [virtual]

This method copies multiple bytes to the output stream. It is assumed the byte are already formatted into a valid XML encoding type (for example, UTF-8).

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

Parameters:

data Array of bytes to copy to the output stream

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void Copy (byte data) [virtual]

This method is used to copy a single byte to the output stream. It first converts the byte to a hex character representation and then copies it to the output buffer.

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

Parameters:

data The byte value to copy

Implements [Asn1XerEncoder](#).virtual void DecrLevel () [virtual]

This method decrements the element nesting level counter.

Implements [Asn1XerEncoder](#).virtual void EncodeBinStrValue (byte[] bits, int nbits) [virtual]

This method encodes XML binary string data

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

Parameters:

bits Bit String to encode
nbits Number of bits to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeByte (byte data) [virtual]

This method is used to encode a single byte to the output stream. It first converts the byte to a hex character representation and then copies it to the output buffer.

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

Parameters:

data The byte value to copy

Implements [Asn1XerEncoder](#).virtual void EncodeData (System.String data) [virtual]

This method encodes XML string data

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

Parameters:

data String value to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeEmptyElement (System.String elemName, System.String attribute) [virtual]

This method encodes an XML empty element tag

Parameters:

elemName The name of element.

attribute The name and value of attribute.

Implements [Asn1XerEncoder](#).virtual void EncodeEmptyElement (System.String elemName) [virtual]

This method encodes an XML empty element tag.

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

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeEndDocument () [virtual]

This method encodes standard trailer information at the end of the XML document.

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

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeEndElement (System.String elemName) [virtual]

This method encodes an XML end element tag.

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

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).Reimplemented in [Asn1XmlOutputStream](#).virtual void EncodeHexStrValue (byte[] data) [virtual]

This method encodes XML hexadecimal string data

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

Parameters:

data Data to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String attribute) [virtual]

This method encodes an XML named value (with start and end tags)

Parameters:

valueName The named value.

elemName The name of element.

attribute The name and value of attribute.

Implements [Asn1XerEncoder](#).virtual void EncodeNamedValue (System.String valueName, System.String elemName) [virtual]

This method encodes an XML named value (with start and end tags).

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

Parameters:

valueName The named value.

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeNamedValueElement (System.String elemName) [virtual]

This method encodes an XML named value element tag.

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

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).Reimplemented in [Asn1XmlOutputStream](#).virtual void EncodeObjectId (int[] data) [virtual]

This method encodes XML Object Identifiers and Relative OIDs data

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

Parameters:

data Object's identifiers to encode

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeRealValue (double data, System.String elemName, System.String attribute) [virtual]

This method encodes an XML REAL (double) value (with start and end tags).

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

Parameters:

data The value to be encoded.

elemName The name of element. If null, then start and end tags won't be encoded.

attribute The name and value of attribute.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeRealValue (double *data*, System.String *elemName*) [virtual]

This method encodes an XML REAL (double) value (with start and end tags).

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

Parameters:

data The value to be encoded.

elemName The name of element. If null, then start and end tags won't be encoded.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeStartDocument () [virtual]

This method encodes standard header information at the beginning of the XML document.

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

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeStartElement (System.String *elemName*, System.String *attribute*) [virtual]

This method encodes an XML start element tag with attribute.

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

Parameters:

elemName The name of element.

attribute The name of attribute.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void EncodeStartElement (System.String *elemName*) [virtual]

This method encodes an XML start element tag.

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

Parameters:

elemName The name of element.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void IncrLevel () [virtual]

This method increments the element nesting level counter.

Implements [Asn1XerEncoder](#).virtual void Indent () [virtual]

This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the output stream.

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

Exceptions:

Asn1Exception Thrown, if operation is failed.

Implements [Asn1XerEncoder](#).virtual void Write (System.String data) [virtual]

This method copies a character string to the output stream.

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

Parameters:

data The string value to copy

Exceptions:

Asn1Exception Thrown, if operation is failed.

Property Documentation

virtual bool Canonical [set]

Sets the canonical encoding rule.

Value: true if canonical encoding; otherwise false .

virtual int State [set]

Sets the state of the buffer.

Value: buffer stat

Implements [Asn1XerEncoder](#).

Asn1XerSaxHandler Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerSaxHandlerInherits [XmlSaxDefaultHandler](#).

Inherited by [Asn1XerOpenType.SaxHandler](#).

Detailed Description

This class extends the DefaultHandler SAX handler class to add items specific to ASN.1 XER encoding.

Public Member Functions

- override void [Error](#) (System.Xml.XmlException exception)
- override void [FatalError](#) (System.Xml.XmlException exception)

- virtual void [Init](#) (int startLevel)
- override void [Warning](#) (System.Xml.XmlException exception)

Protected Member Functions

- internal [Asn1XerSaxHandler](#) ()

Protected Attributes

- internal int [mCurrElemID](#)
- internal int [mCurrState](#)
- internal int [mLevel](#)
- internal int [mStartLevel](#)
- internal readonly int [XERDATA](#) = 2
- internal readonly int [XEREND](#) = 3
- internal readonly int [XERINIT](#) = 0
- internal readonly int [XERSTART](#) = 1
- internal readonly int [XERUNKNOWN](#) = - 1

Properties

- virtual bool [Complete](#)
- virtual int [State](#)

Constructor & Destructor Documentation

internal [Asn1XerSaxHandler](#) () [protected]

The default constructor creates an XER SAX parser instance.

Member Function Documentation

override void [Error](#) (System.Xml.XmlException *exception*) [virtual]

This method manage when an error exception occurs in the parsing process

Parameters:

exception The exception throws by the parser

Exceptions:

System.Xml.XmlException The error exception

Reimplemented from [XmlSaxDefaultHandler](#).override void [FatalError](#) (System.Xml.XmlException *exception*) [virtual]

This method manage when a fatal error exception occurs in the parsing process

Parameters:

exception The exception Throws by the parser

Exceptions:

System.Xml.XmlException The error exception

Reimplemented from [XmlSaxDefaultHandler](#).virtual void [Init](#) (int *startLevel*) [virtual]

This method initializes the this class member variables.

Parameters:

startLevel The XER level to begin

override void Warning (System.Xml.XmlException exception) [virtual]

This method manage when a warning exception occurs in the parsing process

Parameters:

exception The exception Throws by the parser

Exceptions:

System.Xml.XmlException The warning exception

Reimplemented from [XmlSaxDefaultHandler](#).

Member Data Documentation

internal int [mCurrElemID](#) [protected]

Variable holds the current element ID

internal int [mCurrState](#) [protected]

Variable holds the current XER processing state

internal int [mLevel](#) [protected]

Variable holds the start and current level

internal int [mStartLevel](#) [protected]

Variable holds the start and current level

internal readonly int [XERDATA](#) = 2 [protected]

XER characters (data) state

internal readonly int [XEREND](#) = 3 [protected]

XER end element state

internal readonly int [XERINIT](#) = 0 [protected]

XER initialization stat

internal readonly int [XERSTART](#) = 1 [protected]

XER start element state

internal readonly int [XERUNKNOWN](#) = - 1 [protected]

XER unknown stat

Property Documentation

virtual bool Complete [get]

Gets the status of the current element parsing process is complete.

Value: true if found; otherwise false

virtual int State [get]

Gets the current state of the event handler.

Value: current stat

Asn1XerUtil Class Reference

Com::Objsys::Asn1::Runtime::Asn1XerUtil

Detailed Description

This class contains some general purpose static utility functions for XER encoding or decoding.

Static Public Member Functions

- static void [EncodeReal](#) ([Asn1XerEncoder](#) buffer, double value, System.String elemName)
- static System.String [NormalizedRealValueToString](#) (double value, bool isXer)

Member Function Documentation

static void EncodeReal ([Asn1XerEncoder](#) *buffer*, double *value*, System.String *elemName*)
[static]

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

Parameters:

buffer Encode message buffer object
value Value to be encoded.
elemName Element name

static System.String NormalizedRealValueToString (double *value*, bool *isXer*) [static]

This method will return a string representation of the normalized REAL value. The output format is value format [+|-]X.XXXE[-]XXX as defined in X.693.

Parameters:

value value to be normalized and stringified.
isXer true, if this value is being encoded for XER.

Returns:

Stringified representation of the value

Asn1XmlEncodeBuffer Class Reference

Com::ObjSys::Asn1::Runtime::Asn1XmlEncodeBufferInherits [Asn1XerEncodeBuffer](#).

Detailed Description

This class handles the encoding of ASN.1 messages as specified in the XML Encoding (non-XER) by the XML schema standard (generated by asn2xsd).

Public Member Functions

- [Asn1XmlEncodeBuffer](#) (bool canonical, int sizeIncrement)
 - [Asn1XmlEncodeBuffer](#) (bool canonical)
 - [Asn1XmlEncodeBuffer](#) ()
 - override void [EncodeEndElement](#) (System.String elemName)
 - override void [EncodeNamedValueElement](#) (System.String valueName)
-

Constructor & Destructor Documentation

[Asn1XmlEncodeBuffer](#) ()

The default constructor creates an XML encode buffer object with the default size increment and canonical set to false.

[Asn1XmlEncodeBuffer](#) (bool *canonical*)

The parameterized constructor creates an XML encode buffer object with default size increment and canonical set to the given values.

Parameters:

canonical Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.

[Asn1XmlEncodeBuffer](#) (bool *canonical*, int *sizeIncrement*)

The parameterized constructor creates an XML encode buffer object with size increment and canonical set to the given values.

Parameters:

canonical Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.

sizeIncrement The initial size in bytes of an encode buffer. If the buffer becomes full, it will be expanded by this amount. If this parameter is set to zero, the default increment will be used.

Member Function Documentation

override void [EncodeEndElement](#) (System.String *elemName*) [virtual]

This method encodes an XML end element tag

Parameters:

elemName The name of element, as String.

Reimplemented from [Asn1XerEncodeBuffer](#).override void EncodeNamedValueElement (System.String *valueName*) [virtual]

This method encodes an XML named value element tag

Parameters:

valueName The named value, as String.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1XerEncodeBuffer](#).

Asn1XmlOutputStream Class Reference

Com::Objsys::Asn1::Runtime::Asn1XmlOutputStreamInherits [Asn1XerOutputStream](#).

Detailed Description

This class implements the output stream to encode ASN.1 messages as specified in the XML Encoding by the XML schema standard(generated by asn2xsd). A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Public Member Functions

- [Asn1XmlOutputStream](#) (System.IO.Stream os, bool canonical, int bufSize)
 - [Asn1XmlOutputStream](#) (System.IO.Stream os, int bufSize)
 - [Asn1XmlOutputStream](#) (System.IO.Stream os)
 - override void [EncodeEndElement](#) (System.String elemName)
 - override void [EncodeNamedValueElement](#) (System.String valueName)
-

Constructor & Destructor Documentation

[Asn1XmlOutputStream](#) (System.IO.Stream os)

This constructor creates a buffered XML output stream object with default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.

[Asn1XmlOutputStream](#) (System.IO.Stream os, int *bufSize*)

This constructor creates a buffered XML output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.
bufSize The buffer size. If it is 0 then the output stream is used as unbuffered.

[Asn1XmlOutputStream](#) (System.IO.Stream *os*, bool *canonical*, int *bufSize*)

This constructor creates a buffered XML output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream.

Parameters:

os The underlying System.IO.Stream object.
canonical Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.
bufSize The buffer size. If it is 0 then the output stream is used as unbuffered.

Member Function Documentation**override void EncodeEndElement (System.String *elemName*) [virtual]**

This method encodes an XML end element tag

Parameters:

elemName The name of element , as String.

Reimplemented from [Asn1XerOutputStream](#).override void EncodeNamedValueElement (System.String *valueName*) [virtual]

This method encodes an XML named value element tag. Also throws C# Exceptions, If I/O error occurs.

Parameters:

valueName The named value, as String.

Exceptions:

Asn1Exception Thrown, if operation is failed.

Reimplemented from [Asn1XerOutputStream](#).

Asn1XmlUtil Class Reference

Com::Objsys::Asn1::Runtime::Asn1XmlUtil

Detailed Description

This class contains some general purpose static utility functions for XML encoding or decoding.

Static Public Member Functions

- static void [EncodeReal](#) ([Asn1XerEncoder](#) buffer, double value, System.String elemName, System.String attribute)
 - static double [GetMinusZero](#) ()
 - static bool [IsMinusZero](#) (double value)
-

Member Function Documentation

static void EncodeReal ([Asn1XerEncoder](#) *buffer*, double *value*, System.String *elemName*, System.String *attribute*) [static]

This method encodes an ASN.1 real value using the XML encoding (non-XER).

Parameters:

buffer Encode message buffer object
value Value to be encoded.
elemName Element name
attribute Attribute name and value

static double GetMinusZero () [static]

This method returns double value for "minus zero" (-0) special XML value.

Returns:

double value for "-0".

static bool IsMinusZero (double *value*) [static]

This method will return true, if value is "minus zero" (-0) special XML value.

Parameters:

value value to test

Returns:

true, if this value is "-0".

XmlAttributes Class Reference

Com::Objsys::Asn1::Runtime::XmlAttributes

Detailed Description

This class will manage all the parsing operations emulating the SAX parser behavior

Public Member Functions

- virtual void [Add](#) (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)
- virtual void [Clear](#) ()
- virtual System.String [GetFullName](#) (int index)
- virtual int [GetIndex](#) (System.String Uri, System.String Lname)
- virtual int [GetIndex](#) (System.String Qname)
- virtual int [GetLength](#) ()
- virtual System.String [GetLocalName](#) (int index)
- virtual System.String [GetType](#) (System.String Uri, System.String Lname)
- virtual System.String [GetType](#) (System.String Qname)
- virtual System.String [GetType](#) (int index)
- virtual System.String [GetURI](#) (int index)

- virtual System.String [GetValue](#) (System.String Uri, System.String Lname)
- virtual System.String [GetValue](#) (System.String Qname)
- virtual System.String [GetValue](#) (int index)
- virtual void [RemoveAttribute](#) (System.String indexName)
- virtual void [RemoveAttribute](#) (int index)
- virtual void [SetAttribute](#) (int index, System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)
- virtual void [SetAttributes](#) ([XmlAttributes](#) Source)
- virtual void [SetFullName](#) (int index, System.String FullName)
- virtual void [SetLocalName](#) (int index, System.String LocalName)
- virtual void [SetType](#) (int index, System.String Type)
- virtual void [SetURI](#) (int index, System.String URI)
- virtual void [SetValue](#) (int index, System.String Value)
- [XmlAttributes](#) ([XmlAttributes](#) arrayList)
- [XmlAttributes](#) ()

Classes

- class [XmlAttribute](#)

Constructor & Destructor Documentation

[XmlAttributes](#) ()

Builds a new instance of [XmlAttributes](#).

[XmlAttributes](#) ([XmlAttributes](#) arrayList)

Creates a new instance of [XmlAttributes](#) from an ArrayList of [XmlAttribute](#) class.

Parameters:

arrayList An ArraList of [XmlAttribute](#) class instances.

Returns:

A new instance of [XmlAttributes](#)

Member Function Documentation

virtual void Add (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value) [virtual]

Adds a new attribute element to the given [XmlAttributes](#) instance.

Parameters:

Uri The Uri of the attribute to be added.

Lname The Local name of the attribute to be added.

Qname The Long(qualify) name of the attribute to be added.

Type The type of the attribute to be added.

Value The value of the attribute to be added.

virtual void Clear () [virtual]

Clears the list of attributes in the given AttributesSupport instance.

virtual System.String GetFullName (int *index*) [virtual]

Returns the qualified name of the attribute in the given [XmlAttributes](#) instance that indicates the given index.

Parameters:

index The attribute index.

Returns:

The qualified name of the attribute indicated by the index or null if the index is out of bounds.

virtual int GetIndex (System.String *Uri*, System.String *Lname*) [virtual]

Obtains the index of an attribute of the AttributeSupport from its namespace URI and its localname.

Parameters:

Uri The namespace URI of the attribute to search.

Lname The local name of the attribute to search.

Returns:

An zero-based index of the attribute if it is found, otherwise it returns -1.

virtual int GetIndex (System.String *Qname*) [virtual]

Obtains the index of an attribute of the AttributeSupport from its qualified (long) name.

Parameters:

Qname The qualified name of the attribute to search.

Returns:

An zero-based index of the attribute if it is found, otherwise it returns -1.

virtual int GetLength () [virtual]

Returns the number of attributes saved in the [XmlAttributes](#) instance.

Returns:

The number of elements in the given [XmlAttributes](#) instance.

virtual System.String GetLocalName (int *index*) [virtual]

Returns the local name of the attribute in the given [XmlAttributes](#) instance that indicates the given index.

Parameters:

index The attribute index.

Returns:

The local name of the attribute indicated by the index or null if the index is out of bounds.

virtual System.String GetType (System.String *Uri*, System.String *Lname*) [virtual]

Returns the type of the Attribute that match with the given namespace URI and local name.

Parameters:

Uri The namespace URI of the attribute to search.

Lname The local name of the attribute to search.

Returns:

The type of the attribute if it exist otherwise returns null.

virtual System.String GetType (System.String Qname) [virtual]

Returns the type of the Attribute that match with the given qualified name.

Parameters:

Qname The qualified name of the attribute to search.

Returns:

The type of the attribute if it exist otherwise returns null.

virtual System.String GetType (int index) [virtual]

Returns the type of the attribute in the given [XmlAttributes](#) instance that indicates the given index.

Parameters:

index The attribute index.

Returns:

The type of the attribute indicated by the index or null if the index is out of bounds.

virtual System.String GetURI (int index) [virtual]

Returns the namespace URI of the attribute in the given [XmlAttributes](#) instance that indicates the given index.

Parameters:

index The attribute index.

Returns:

The namespace URI of the attribute indicated by the index or null if the index is out of bounds.

virtual System.String GetValue (System.String Uri, System.String Lname) [virtual]

Returns the value of the Attribute that match with the given namespace URI and local name.

Parameters:

Uri The namespace URI of the attribute to search.

Lname The local name of the attribute to search.

Returns:

The value of the attribute if it exist otherwise returns null.

virtual System.String GetValue (System.String Qname) [virtual]

Returns the value of the Attribute that match with the given qualified name.

Parameters:

Qname The qualified name of the attribute to search.

Returns:

The value of the attribute if it exist otherwise returns null.

virtual System.String GetValue (int index) [virtual]

Returns the value of the attribute in the given [XmlAttributes](#) instance that indicates the given index.

Parameters:

index The attribute index.

Returns:

The value of the attribute indicated by the index or null if the index is out of bounds.

virtual void RemoveAttribute (System.String *indexName*) [virtual]

This method eliminates the [XmlAttribute](#) instance in the specified index.

Parameters:

indexName The index name of the attribute.

virtual void RemoveAttribute (int *index*) [virtual]

This method eliminates the [XmlAttribute](#) instance at the specified index.

Parameters:

index The index of the attribute.

virtual void SetAttribute (int *index*, System.String *Uri*, System.String *Lname*, System.String *Qname*, System.String *Type*, System.String *Value*) [virtual]

Replaces an [XmlAttribute](#) in the given [XmlAttributes](#) instance.

Parameters:

index The index of the attribute.

Uri The namespace URI of the new [XmlAttribute](#).

Lname The local name of the new [XmlAttribute](#).

Qname The namespace URI of the new [XmlAttribute](#).

Type The type of the new [XmlAttribute](#).

Value The value of the new [XmlAttribute](#).

virtual void SetAttributes ([XmlAttributes](#) *Source*) [virtual]

Replaces all the list of [XmlAttribute](#) of the given [XmlAttributes](#) instance.

Parameters:

Source The source [XmlAttributes](#) instance.

virtual void SetFullName (int *index*, System.String *FullName*) [virtual]

Modifies the qualified name of the attribute in the given [XmlAttributes](#) instance.

Parameters:

index The attribute index.

FullName The new qualified name for the attribute.

virtual void SetLocalName (int *index*, System.String *LocalName*) [virtual]

Modifies the local name of the attribute in the given [XmlAttributes](#) instance.

Parameters:

index The attribute index.

LocalName The new Local name for the attribute.

virtual void SetType (int *index*, System.String *Type*) [virtual]

Modifies the type of the attribute in the given [XmlAttributes](#) instance.

Parameters:

index The attribute index.

Type The new type for the attribute.

virtual void SetURI (int *index*, System.String *URI*) [virtual]

Modifies the namespace URI of the attribute in the given [XmlAttributes](#) instance.

Parameters:

index The attribute index.

URI The new namespace URI for the attribute.

virtual void SetValue (int *index*, System.String *Value*) [virtual]

Modifies the value of the attribute in the given [XmlAttributes](#) instance.

Parameters:

index The attribute index.

Value The new value for the attribute.

XmlAttributes.XmlAttribute Class Reference

Com::Objsys::Asn1::Runtime::XmlAttributes::XmlAttribute

Detailed Description

This class is created to save the information of each attributes in the [XmlAttributes](#).

Public Member Functions

- [XmlAttribute](#) (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)

Public Attributes

- System.String [att_fullName](#)
- System.String [att_localName](#)
- System.String [att_type](#)
- System.String [att_URI](#)
- System.String [att_value](#)

Constructor & Destructor Documentation

[XmlAttribute](#) (System.String *Uri*, System.String *Lname*, System.String *Qname*, System.String *Type*, System.String *Value*)

This is the constructor of the [XmlAttribute](#)

Parameters:

Uri The namespace URI of the attribute

Lname The local name of the attribute

Qname The long(Qualify) name of attribute

Type The type of the attribute

Value The value of the attribute

Member Data Documentation

System.String [att fullName](#)

Variable holds attribte full name (namespace + local name)

System.String [att localName](#)

Variable holds attribte local name

System.String [att type](#)

Variable holds attribte type

System.String [att URI](#)

Variable holds attribte namespace

System.String [att value](#)

Variable holds attribte value

XmlSaxContentHandler Interface Reference

Com::Objsys::Asn1::Runtime::XmlSaxContentHandlerInherited by [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).

Detailed Description

This interface will manage the Content events of a XML document.

Public Member Functions

- void [Characters](#) (char[] ch, int start, int length)
- void [EndDocument](#) ()
- void [EndElement](#) (System.String namespaceURI, System.String localName, System.String qName)
- void [EndPrefixMapping](#) (System.String prefix)
- void [IgnorableWhitespace](#) (char[] Ch, int Start, int Length)
- void [ProcessingInstruction](#) (System.String target, System.String data)
- void [SetDocumentLocator](#) ([XmlSaxLocator](#) locator)
- void [SkippedEntity](#) (System.String name)
- void [StartDocument](#) ()
- void [StartElement](#) (System.String namespaceURI, System.String localName, System.String qName, [XmlAttributes](#) atts)
- void [StartPrefixMapping](#) (System.String prefix, System.String uri)

Member Function Documentation

void Characters (char[] *ch*, int *start*, int *length*)

This method manage the notification when Characters elements were found.

Parameters:

ch The array with the characters found.
start The index of the first position of the characters found.
length Specify how many characters must be read from the array.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void EndDocument ()

This method manage the notification when the end document node were found.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void EndElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*)

This method manage the notification when the end element node was found.

Parameters:

namespaceURI The namespace URI of the element.
localName The local name of the element.
qName The long (qualified) name of the element.

Implemented in [Asn1XerOpenType.SaxHandler](#), [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void EndPrefixMapping (System.String *prefix*)

This method manage the event when an area of expecific URI prefix was ended.

Parameters:

prefix The prefix that ends.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void IgnorableWhitespace (char[] *Ch*, int *Start*, int *Length*)

This method manage the event when a ignorable whitespace node was found.

Parameters:

Ch The array with the ignorable whitespaces.
Start The index in the array with the ignorable whitespace.
Length The length of the whitespaces.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void ProcessingInstruction (System.String *target*, System.String *data*)

This method manage the event when a processing instruction was found.

Parameters:

target The processing instruction target.
data The processing instruction data.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void SetDocumentLocator ([XmlSaxLocator](#) *locator*)

This method is not supported, it is included for compatibility.

Parameters:

locator A [XmlSaxLocator](#) object that can return the location of any events into the XML document

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void SkippedEntity (System.String name)

This method manage the event when a skipped entity was found.

Parameters:

name The name of the skipped entity.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void StartDocument ()

This method manage the event when a start document node was found.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void StartElement (System.String namespaceURI, System.String localName, System.String qName, [XmlAttributes](#) atts)

This method manage the event when a start element node was found.

Parameters:

namespaceURI The namespace uri of the element tag.

localName The local name of the element.

qName The long (qualified) name of the element.

atts The list of attributes of the element.

Implemented in [Asn1XerOpenType.SaxHandler](#), [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).void StartPrefixMapping (System.String prefix, System.String uri)

This methods indicates the start of a prefix area in the XML document.

Parameters:

prefix The prefix of the area.

uri The namespace URI of the prefix area.

Implemented in [XmlSaxDefaultHandler](#), and [XmlSaxParserAdapter](#).

XmlSaxDefaultHandler Class Reference

Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandlerInherits [XmlSaxContentHandler](#), [XmlSaxErrorHandler](#), and [XmlSaxEntityResolver](#).

Inherited by [Asn1XerSaxHandler](#).

Detailed Description

This class provides the base implementation for the management of XML documents parsing.

Public Member Functions

- virtual void [Characters](#) (char[] ch, int start, int length)
- virtual void [EndDocument](#) ()
- virtual void [EndElement](#) (System.String ns, System.String localName, System.String qName)
- virtual void [EndPrefixMapping](#) (System.String prefix)
- virtual void [Error](#) (System.Xml.XmlException exception)

- virtual void [FatalError](#) (System.Xml.XmlException exception)
 - virtual void [IgnorableWhitespace](#) (char[] chars, int start, int length)
 - virtual void [ProcessingInstruction](#) (System.String target, System.String data)
 - virtual [XmlSource ResolveEntity](#) (System.String publicId, System.String systemId)
 - virtual void [SetDocumentLocator](#) ([XmlSaxLocator](#) locator)
 - virtual void [SkippedEntity](#) (System.String name)
 - virtual void [StartDocument](#) ()
 - virtual void [StartElement](#) (System.String ns, System.String localName, System.String qName, [XmlAttributes](#) attributes)
 - virtual void [StartPrefixMapping](#) (System.String prefix, System.String uri)
 - virtual void [Warning](#) (System.Xml.XmlException exception)
-

Member Function Documentation

virtual void Characters (char[] *ch*, int *start*, int *length*) [virtual]

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

Implements [XmlSaxContentHandler](#).virtual void EndDocument () [virtual]

This method manage the notification when the end document node were found

Implements [XmlSaxContentHandler](#).virtual void EndElement (System.String *ns*, System.String *localName*, System.String *qName*) [virtual]

This method manage the notification when the end element node were found

Parameters:

- ns* The namespace of the element
- localName* The local name of the element
- qName* The long name (qualify name) of the element

Implements [XmlSaxContentHandler](#).Reimplemented in [Asn1XerOpenType.SaxHandler](#).virtual void EndPrefixMapping (System.String *prefix*) [virtual]

This method manage the event when an area of expecific URI prefix was ended.

Parameters:

- prefix* The prefix that ends

Implements [XmlSaxContentHandler](#).virtual void Error (System.Xml.XmlException *exception*) [virtual]

This method manage when an error exception occurs in the parsing process

Parameters:

- exception* The exception thrown by the parser

Implements [XmlSaxErrorHandler](#). Reimplemented in [Asn1XerSaxHandler](#). virtual void FatalError (System.Xml.XmlException exception) [virtual]

This method manage when a fatal error exception occurs in the parsing process

Parameters:

exception The exception Thrown by the parser

Implements [XmlSaxErrorHandler](#). Reimplemented in [Asn1XerSaxHandler](#). virtual void IgnorableWhitespace (char[] chars, int start, int length) [virtual]

This method manage the event when a ignorable whitespace node were found

Parameters:

chars The array with the ignorable whitespaces

start The array offset at the ignorable whitespace

length The length of the whitespaces

Implements [XmlSaxContentHandler](#). virtual void ProcessingInstruction (System.String target, System.String data) [virtual]

This method manage the event when a processing instruction were found

Parameters:

target The processing instruction target

data The processing instruction data

Implements [XmlSaxContentHandler](#). virtual [XmlSource](#) ResolveEntity (System.String publicId, System.String systemId) [virtual]

Allow the application to resolve external entities.

Parameters:

publicId The public identifier of the external entity being referenced, or null if none was supplied.

systemId The system identifier of the external entity being referenced.

Returns:

A XmlSourceSupport object describing the new input source, or null to request that the parser open a regular URI connection to the system identifier.

Implements [XmlSaxEntityResolver](#). virtual void SetDocumentLocator ([XmlSaxLocator](#) locator) [virtual]

This method is not supported, is include for compatibility

Parameters:

locator A [XmlSaxLocator](#) object that can return the location of any events into the XML document

Implements [XmlSaxContentHandler](#). virtual void SkippedEntity (System.String name) [virtual]

This method manage the event when a skipped entity were found

Parameters:

name The name of the skipped entity

Implements [XmlSaxContentHandler](#). virtual void StartDocument () [virtual]

This method manage the event when a start document node were found

Implements [XmlSaxContentHandler](#).virtual void StartElement (System.String ns, System.String localName, System.String qName, [XmlAttributes](#) attributes) [virtual]

This method manage the event when a start element node were found

Parameters:

ns The namespace uri of the element tag
localName The local name of the element
qName The Qualify (long) name of the element
attributes The list of attributes of the element

Implements [XmlSaxContentHandler](#).Reimplemented in [Asn1XerOpenType.SaxHandler](#).virtual void StartPrefixMapping (System.String prefix, System.String uri) [virtual]

This methods indicates the start of a prefix area in the XML document.

Parameters:

prefix The prefix of the area
uri The namespace uri of the prefix area

Implements [XmlSaxContentHandler](#).virtual void Warning (System.Xml.XmlException exception) [virtual]

This method manage when a warning exception occurs in the parsing process

Parameters:

exception The exception Thrown by the parser

Implements [XmlSaxErrorHandler](#).Reimplemented in [Asn1XerSaxHandler](#).

XmlSaxEntityResolver Interface Reference

Com::Objsys::Asn1::Runtime::XmlSaxEntityResolverInherited by [XmlSaxDefaultHandler](#).

Detailed Description

Basic interface for resolving entities.

Public Member Functions

- [XmlSource ResolveEntity](#) (System.String publicId, System.String systemId)
-

Member Function Documentation

[XmlSource](#) ResolveEntity (System.String publicId, System.String systemId)

Allow the application to resolve external entities.

Parameters:

publicId The public identifier of the external entity being referenced, or null if none was supplied.
systemId The system identifier of the external entity being referenced.

Returns:

A XmlSourceSupport object describing the new input source, or null to request that the parser open a regular URI connection to the system identifier.

Implemented in [XmlSaxDefaultHandler](#).

XmlSaxErrorHandler Interface Reference

Com::Objsys::Asn1::Runtime::XmlSaxErrorHandlerInherited by [XmlSaxDefaultHandler](#).

Detailed Description

This interface will manage errors during the parsing of a XML document.

Public Member Functions

- void [Error](#) (System.Xml.XmlException exception)
 - void [FatalError](#) (System.Xml.XmlException exception)
 - void [Warning](#) (System.Xml.XmlException exception)
-

Member Function Documentation

void Error (System.Xml.XmlException exception)

This method manage an error exception occurred during the parsing process.

Parameters:

exception The exception thrown by the parser.

Implemented in [Asn1XerSaxHandler](#), and [XmlSaxDefaultHandler](#).void FatalError (System.Xml.XmlException exception)

This method manage a fatal error exception occurred during the parsing process.

Parameters:

exception The exception thrown by the parser.

Implemented in [Asn1XerSaxHandler](#), and [XmlSaxDefaultHandler](#).void Warning (System.Xml.XmlException exception)

This method manage a warning exception occurred during the parsing process.

Parameters:

exception The exception thrown by the parser.

Implemented in [Asn1XerSaxHandler](#), and [XmlSaxDefaultHandler](#).

XmlSaxLexicalHandler Interface Reference

Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler

Detailed Description

This interface will manage the Content events of a XML document.

Public Member Functions

- void [Comment](#) (char[] ch, int start, int length)
- void [EndCDATA](#) ()
- void [EndDTD](#) ()
- void [EndEntity](#) (System.String name)
- void [StartCDATA](#) ()
- void [StartDTD](#) (System.String name, System.String publicId, System.String systemId)
- void [StartEntity](#) (System.String name)

Member Function Documentation

void Comment (char[] *ch*, int *start*, int *length*)

This method manage the notification when Characters elements were found.

Parameters:

ch The array with the characters found.

start The index of the first position of the characters found.

length Specify how many characters must be read from the array.

void EndCDATA ()

This method manage the notification when the end of a CDATA section were found.

void EndDTD ()

This method manage the notification when the end of DTD declarations were found.

void EndEntity (System.String *name*)

This method report the end of an entity.

Parameters:

name The name of the entity that is ending.

void StartCDATA ()

This method manage the notification when the start of a CDATA section were found.

void StartDTD (System.String *name*, System.String *publicId*, System.String *systemId*)

This method manage the notification when the start of DTD declarations were found.

Parameters:

name The name of the DTD entity.

publicId The public identifier.

systemId The system identifier.

void StartEntity (System.String name)

This method report the start of an entity.

Parameters:

name The name of the entity that is ending.

XmlSaxLocator Interface Reference

Com::Objsys::Asn1::Runtime::XmlSaxLocatorInherited by [XmlSaxLocatorImpl](#).

Detailed Description

This interface is created to emulate the SAX Locator interface behavior.

Public Member Functions

- int [GetColumnNumber](#) ()
 - int [GetLineNumber](#) ()
 - System.String [GetPublicId](#) ()
 - System.String [GetSystemId](#) ()
-

Member Function Documentation

int GetColumnNumber ()

This method return the column number where the current document event ends.

Returns:

The column number where the current document event ends.

Implemented in [XmlSaxLocatorImpl](#).int GetLineNumber ()

This method return the line number where the current document event ends.

Returns:

The line number where the current document event ends.

Implemented in [XmlSaxLocatorImpl](#).System.String GetPublicId ()

This method is not supported, it is included for compatibility.

Returns:

The saved public identifier.

Implemented in [XmlSaxLocatorImpl](#).System.String GetSystemId ()

This method is not supported, it is included for compatibility.

Returns:

The saved system identifier.

Implemented in [XmlSaxLocatorImpl](#).

XmlSaxLocatorImpl Class Reference

Com::Objsys::Asn1::Runtime::XmlSaxLocatorImplInherits [XmlSaxLocator](#).

Detailed Description

This class is created for emulates the SAX LocatorImpl behaviors.

Public Member Functions

- virtual int [GetColumnNumber](#) ()
 - virtual int [GetLineNumber](#) ()
 - virtual System.String [GetPublicId](#) ()
 - virtual System.String [GetSystemId](#) ()
 - virtual void [SetColumnNumber](#) (int columnNumber)
 - virtual void [SetLineNumber](#) (int lineNumber)
 - virtual void [SetPublicId](#) (System.String publicId)
 - virtual void [SetSystemId](#) (System.String systemId)
 - [XmlSaxLocatorImpl](#) ([XmlSaxLocator](#) locator)
 - [XmlSaxLocatorImpl](#) ()
-

Constructor & Destructor Documentation

[XmlSaxLocatorImpl](#) ()

This method returns a new instance of 'XmlSaxLocatorImpl'.

Returns:

A new 'XmlSaxLocatorImpl' instance.

[XmlSaxLocatorImpl](#) ([XmlSaxLocator](#) *locator*)

This method returns a new instance of 'XmlSaxLocatorImpl'. Create a persistent copy of the current state of a locator.

Parameters:

locator The current state of a locator.

Returns:

A new 'XmlSaxLocatorImpl' instance.

Member Function Documentation

virtual int GetColumnNumber () [virtual]

Return the saved column number.

Returns:

The saved column number.

Implements [XmlSaxLocator](#).virtual int GetLineNumber () [virtual]

Return the saved line number.

Returns:

The saved line number.

Implements [XmlSaxLocator](#).virtual System.String GetPublicId () [virtual]

This method is not supported, it is included for compatibility. Return the saved public identifier.

Returns:

The saved public identifier.

Implements [XmlSaxLocator](#).virtual System.String GetSystemId () [virtual]

This method is not supported, it is included for compatibility. Return the saved system identifier.

Returns:

The saved system identifier.

Implements [XmlSaxLocator](#).virtual void SetColumnNumber (int *columnNumber*) [virtual]

Set the column number for this locator.

Parameters:

columnNumber The column number.

virtual void SetLineNumber (int *lineNumber*) [virtual]

Set the line number for this locator.

Parameters:

lineNumber The line number.

virtual void SetPublicId (System.String *publicId*) [virtual]

This method is not supported, it is included for compatibility. Set the public identifier for this locator.

Parameters:

publicId The new public identifier.

virtual void SetSystemId (System.String *systemId*) [virtual]

This method is not supported, it is included for compatibility. Set the system identifier for this locator.

Parameters:

systemId The new system identifier.

XmlSaxParser Class Reference

Com::Obsys::Asn1::Runtime::XmlSaxParserInherited by [XmlSaxParserAdapter](#).

Detailed Description

Emulates the SAX parsers behaviours.

Public Member Functions

- virtual [XmlSaxContentHandler](#) [GetContentHandler](#) ()
- virtual [XmlSaxEntityResolver](#) [GetEntityResolver](#) ()
- virtual [XmlSaxErrorHandler](#) [GetErrorHandler](#) ()
- virtual void [Parse](#) ([XmlSource](#) source)
- virtual void [Parse](#) (System.IO.Stream stream, System.String URI)
- virtual void [Parse](#) (System.IO.Stream stream)
- virtual void [Parse](#) (System.String filepath)
- virtual void [Parse](#) (System.IO.FileInfo filepath)
- virtual void [Parse](#) ([XmlSource](#) source, [XmlSaxContentHandler](#) handler)
- virtual void [Parse](#) (System.IO.Stream stream, [XmlSaxContentHandler](#) handler, System.String URI)
- virtual void [Parse](#) (System.IO.Stream stream, [XmlSaxContentHandler](#) handler)
- virtual void [Parse](#) (System.String filepath, [XmlSaxContentHandler](#) handler)
- virtual void [Parse](#) (System.IO.FileInfo filepath, [XmlSaxContentHandler](#) handler)
- virtual void [SetContentHandler](#) ([XmlSaxContentHandler](#) handler)
- virtual void [SetDocumentHandler](#) ([XmlSaxContentHandler](#) handler)
- virtual void [SetEntityResolver](#) ([XmlSaxEntityResolver](#) resolver)
- virtual void [SetErrorHandler](#) ([XmlSaxErrorHandler](#) handler)
- [XmlSaxParser](#) ()

Static Public Member Functions

- static [XmlSaxParser](#) [CloneInstance](#) ([XmlSaxParser](#) instance)
- static [XmlSaxParser](#) [NewInstance](#) ()

Protected Attributes

- [XmlSaxContentHandler](#) [callBackHandler](#)
- [XmlSaxEntityResolver](#) [entityResolver](#)
- [XmlSaxErrorHandler](#) [errorHandler](#)
- [XmlSaxLexicalHandler](#) [lexical](#)
- [XmlSaxLocatorImpl](#) [locator](#)
- bool [namespaceAllowed](#)
- System.String [parserFileName](#)
- System.Xml.XmlTextReader [reader](#)

Properties

- bool [NamespaceAllowed](#)
-

Constructor & Destructor Documentation

[XmlSaxParser](#) ()

Public constructor for the class.

Member Function Documentation

static [XmlSaxParser](#) CloneInstance ([XmlSaxParser](#) *instance*) [static]

Create a clone instance of 'XmlSaxParser'.

Parameters:

instance The [XmlSaxParser](#) instance to be cloned.

Returns:

A clone 'XmlSaxParser' instance.

virtual [XmlSaxContentHandler](#) GetContentHandler () [virtual]

Obtains the object that will handle all the content events.

Returns:

The object that handles the content events.

virtual [XmlSaxEntityResolver](#) GetEntityResolver () [virtual]

Returns the current entity resolver.

Returns:

The current entity resolver, or null if none has been registered.

virtual [XmlSaxErrorHandler](#) GetErrorHandler () [virtual]

Assigns the object that will handle all the error events.

Returns:

The object that handles the error events.

static [XmlSaxParser](#) NewInstance () [static]

Returns a new instance of 'XmlSaxParser'.

Returns:

A new 'XmlSaxParser' instance.

virtual void Parse ([XmlSource](#) *source*) [virtual]

Parses the specified 'XmlSource' and processes the events over the specified handler, and resolves the entities with the specified URI.

Parameters:

source The 'XmlSource' instance with the XML.

virtual void Parse (System.IO.Stream *stream*, System.String *URI*) [virtual]

Parses the specified stream and processes the events over previously specified handler, and resolves the external entities with the specified URI.

Parameters:

stream The stream with the XML.
URI The namespace URI for resolve external entities.

virtual void Parse (System.IO.Stream *stream*) [virtual]

Parses the specified stream and process the events over previously specified handler.

Parameters:

stream The stream with the XML.

virtual void Parse (System.String *filepath*) [virtual]

Parses the specified file path and processes the events over previously specified handler.

Parameters:

filepath The path of the file with the XML.

virtual void Parse (System.IO.FileInfo *filepath*) [virtual]

Parses the specified file and process the events over previously specified handler.

Parameters:

filepath The file with the XML.

virtual void Parse ([XmlSource](#) *source*, [XmlSaxContentHandler](#) *handler*) [virtual]

Parses the specified 'XmlSource' instance and process the events over the specified handler, and resolves the entities with the specified URI.

Parameters:

source The 'XmlSource' that contains the XML.
handler The handler that manages the parser events.

virtual void Parse (System.IO.Stream *stream*, [XmlSaxContentHandler](#) *handler*, System.String *URI*) [virtual]

Parses the specified stream and process the events over the specified handler, and resolves the entities with the specified URI.

Parameters:

stream The stream with the XML.
handler The handler that manage the parser events.
URI The namespace URI for resolve external entities.

virtual void Parse (System.IO.Stream *stream*, [XmlSaxContentHandler](#) *handler*) [virtual]

Parses the specified stream and process the events over the specified handler.

Parameters:

stream The stream with the XML.
handler The handler that manage the parser events.

virtual void Parse (System.String *filepath*, [XmlSaxContentHandler](#) *handler*) [virtual]

Parses the specified file path and process the events over the specified handler.

Parameters:

filepath The path of the file to be used.
handler The handler that manage the parser events.

virtual void Parse (System.IO.FileInfo *filepath*, [XmlSaxContentHandler](#) *handler*) [virtual]

Parses the specified file and process the events over the specified handler.

Parameters:

filepath The file to be used.

handler The handler that manages the parser events.

virtual void SetContentHandler ([XmlSaxContentHandler](#) *handler*) [virtual]

Assigns the object that will handle all the content events.

Parameters:

handler The object that handles the content events.

virtual void SetDocumentHandler ([XmlSaxContentHandler](#) *handler*) [virtual]

Assigns the object that will handle all the document events.

Parameters:

handler The object that handles the content events.

virtual void SetEntityResolver ([XmlSaxEntityResolver](#) *resolver*) [virtual]

Allows an application to register an entity resolver.

Parameters:

resolver The entity resolver.

virtual void SetErrorHandler ([XmlSaxErrorHandler](#) *handler*) [virtual]

Assigns the object that will handle all the error events.

Parameters:

handler The object that handles the errors events.

Member Data Documentation

[XmlSaxContentHandler callbackHandler](#) [protected]

[XmlSaxContentHandler](#) variable manages the Content events of a XML document.

[XmlSaxEntityResolver entityResolver](#) [protected]

[XmlSaxEntityResolver](#) variable for resolving entities

[XmlSaxErrorHandler errorHandler](#) [protected]

[XmlSaxErrorHandler](#) variable manages errors during the parsing of a XML document

[XmlSaxLexicalHandler lexical](#) [protected]

[XmlSaxLexicalHandler](#) variable manages the Content events of a XML document

[XmlSaxLocatorImpl locator](#) [protected]

[XmlSaxLocatorImpl](#) variable to emulates the SAX LocatorImpl behaviors.

bool [namespaceAllowed](#) [protected]

Bool variable manages XML document namespace processing.

System.String [parserFileName](#) [protected]

String variable holds the XER or XML message file name

System.Xml.XmlTextReader [reader](#) [protected]

XmlTextReader variable manages XML document parsing.

Property Documentation

bool NamespaceAllowed [get, set]

Indicates whether the 'XmlSaxParser' allows namespaces.

XmlSaxParserAdapter Class Reference

Com::Objsys::Asn1::Runtime::XmlSaxParserAdapterInherits [XmlSaxParser](#), and [XmlSaxContentHandler](#).

Detailed Description

This class provides the base implementation for the management of XML documents parsing.

Public Member Functions

- virtual void [Characters](#) (char[] ch, int start, int length)
 - virtual void [EndDocument](#) ()
 - virtual void [EndElement](#) (System.String namespaceURI, System.String localName, System.String qName)
 - virtual void [EndPrefixMapping](#) (System.String prefix)
 - virtual void [IgnorableWhitespace](#) (char[] ch, int start, int length)
 - virtual void [ProcessingInstruction](#) (System.String target, System.String data)
 - virtual void [SetDocumentLocator](#) ([XmlSaxLocator](#) locator)
 - virtual void [SkippedEntity](#) (System.String name)
 - virtual void [StartDocument](#) ()
 - virtual void [StartElement](#) (System.String namespaceURI, System.String localName, System.String qName, [XmlAttributes](#) qAtts)
 - virtual void [StartPrefixMapping](#) (System.String prefix, System.String uri)
-

Member Function Documentation

virtual void Characters (char[] *ch*, int *start*, int *length*) [virtual]

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

Implements [XmlSaxContentHandler](#).virtual void EndDocument () [virtual]

This method manage the notification when the end document node were found

Implements [XmlSaxContentHandler](#).virtual void EndElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*) [virtual]

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

Implements [XmlSaxContentHandler](#).virtual void EndPrefixMapping (System.String *prefix*) [virtual]

This method manage the event when an area of expecific URI prefix was ended.

Parameters:

prefix The prefix that ends.

Implements [XmlSaxContentHandler](#).virtual void IgnorableWhitespace (char[] *ch*, int *start*, int *length*) [virtual]

This method manage the event when a ignorable whitespace node were found

Parameters:

ch The array with the ignorable whitespaces
start The index in the array with the ignorable whitespace
length The length of the whitespaces

Implements [XmlSaxContentHandler](#).virtual void ProcessingInstruction (System.String *target*, System.String *data*) [virtual]

This method manage the event when a processing instruction were found

Parameters:

target The processing instruction target
data The processing instruction data

Implements [XmlSaxContentHandler](#).virtual void SetDocumentLocator ([XmlSaxLocator](#) *locator*) [virtual]

Receive an object for locating the origin of events into the XML document

Parameters:

locator A [XmlSaxLocator](#) object that can return the location of any events into the XML document

Implements [XmlSaxContentHandler](#).virtual void SkippedEntity (System.String name) [virtual]

This method manage the event when a skipped entity was found.

Parameters:

name The name of the skipped entity.

Implements [XmlSaxContentHandler](#).virtual void StartDocument () [virtual]

This method manage the event when a start document node were found

Implements [XmlSaxContentHandler](#).virtual void StartElement (System.String namespaceURI, System.String localName, System.String qName, [XmlAttribute](#) qAtts) [virtual]

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
qAtts The list of attributes of the element

Implements [XmlSaxContentHandler](#).virtual void StartPrefixMapping (System.String prefix, System.String uri) [virtual]

This methods indicates the start of a prefix area in the XML document.

Parameters:

prefix The prefix of the area.
uri The namespace URI of the prefix area.

Implements [XmlSaxContentHandler](#).

XmlSource Class Reference

Com::Objsys::Asn1::Runtime::XmlSource

Detailed Description

This class is used to encapsulate a source of Xml code in an single class.

Public Member Functions

- [XmlSource](#) (System.String source)
- [XmlSource](#) (System.IO.StreamReader reader)
- [XmlSource](#) (System.IO.Stream stream)
- [XmlSource](#) ()

Properties

- System.IO.Stream [Bytes](#)
 - System.IO.StreamReader [Characters](#)
 - System.String [Uri](#)
-

Constructor & Destructor Documentation

[XmlSource](#) ()

Constructs an empty [XmlSource](#) instance.

[XmlSource](#) (System.IO.Stream *stream*)

Constructs a [XmlSource](#) instance with the specified source System.IO.Stream.

Parameters:

stream The stream containing the document.

[XmlSource](#) (System.IO.StreamReader *reader*)

Constructs a [XmlSource](#) instance with the specified source System.IO.StreamReader.

Parameters:

reader The reader containing the document.

[XmlSource](#) (System.String *source*)

Construct a [XmlSource](#) instance with the specified source Uri string.

Parameters:

source The source containing the document.

Property Documentation

System.IO.Stream Bytes [get, set]

Represents the source Stream of the [XmlSource](#).

System.IO.StreamReader Characters [get, set]

Represents the source StreamReader of the [XmlSource](#).

System.String Uri [get, set]

Represents the source URI of the [XmlSource](#).

Index

- Add
 - Com::Objsys::Asn1::Runtime::XmlAttributes, 97
- AddCaptureBuffer
 - Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 50
- AddElemName
 - Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 27
 - Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59
- Aligned
 - Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 56
- Asn1BerDecodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 2
- Asn1BerDecodeContext
 - Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 5
- Asn1BerEncodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 7, 8
- Asn1BerInputStream
 - Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 12
- Asn1BerMessageDumpHandler
 - Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler, 13
- Asn1BerOutputStream
 - Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 15
- Asn1CerInputStream
 - Com::Objsys::Asn1::Runtime::Asn1CerInputStream, 18
- Asn1CerOutputStream
 - Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 19
- Asn1DerDecodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1DerDecodeBuffer, 22
- Asn1DerEncodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1DerEncodeBuffer, 23
- Asn1DerInputStream
 - Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 23
- Asn1NotInSetException
 - Com::Objsys::Asn1::Runtime::Asn1NotInSetException, 25
- Asn1PerBitField
 - Com::Objsys::Asn1::Runtime::Asn1PerBitField, 25
- Asn1PerBitFieldPrinter
 - Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 28
- Asn1PerDecodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 31
- Asn1PerDecodeTraceHandler
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler, 36
- Asn1PerEncodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 38
- Asn1PerEncodeTraceHandler
 - Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler, 45
- Asn1PerInputStream
 - Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 46
- Asn1PerOutputStream
 - Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 50
- Asn1PerOutputStreamTraceHandler
 - Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler, 57
- Asn1PerTraceHandler
 - Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 58
- Asn1SetDuplicateException
 - Com::Objsys::Asn1::Runtime::Asn1SetDuplicateException, 61
- Asn1TagMatchFailedException
 - Com::Objsys::Asn1::Runtime::Asn1TagMatchFailedException, 62, 63
- Asn1XerBase64OctetString
 - Com::Objsys::Asn1::Runtime::Asn1XerBase64OctetString, 63, 64
- Asn1XerDecodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer, 65
- Asn1XerElemInfo
 - Com::Objsys::Asn1::Runtime::Asn1XerElemInfo, 66
- Asn1XerEncodeBuffer
 - Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 68
- Asn1XerOpenType

Com::Objsys::Asn1::Runtime::Asn1XerOpen
 Type, 80, 81
 Asn1XerOutputStream
 Com::Objsys::Asn1::Runtime::Asn1XerOutput
 tStream, 84
 Asn1XerSaxHandler
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHa
 ndler, 90
 Asn1XmlEncodeBuffer
 Com::Objsys::Asn1::Runtime::Asn1XmlEnco
 deBuffer, 93
 Asn1XmlOutputStream
 Com::Objsys::Asn1::Runtime::Asn1XmlOutp
 utStream, 94, 95
 att_fullName
 Com::Objsys::Asn1::Runtime::XmlAttributes:
 :XmlAttribute, 102
 att_localName
 Com::Objsys::Asn1::Runtime::XmlAttributes:
 :XmlAttribute, 102
 att_type
 Com::Objsys::Asn1::Runtime::XmlAttributes:
 :XmlAttribute, 102
 att_URI
 Com::Objsys::Asn1::Runtime::XmlAttributes:
 :XmlAttribute, 102
 att_value
 Com::Objsys::Asn1::Runtime::XmlAttributes:
 :XmlAttribute, 102
 Available
 Com::Objsys::Asn1::Runtime::Asn1BerInputS
 tream, 12
 Com::Objsys::Asn1::Runtime::Asn1DerInputS
 tream, 24
 Com::Objsys::Asn1::Runtime::Asn1PerInputS
 tream, 46
 BinDump
 Com::Objsys::Asn1::Runtime::Asn1BerEncod
 eBuffer, 8
 Com::Objsys::Asn1::Runtime::Asn1PerDecod
 eBuffer, 31
 Com::Objsys::Asn1::Runtime::Asn1PerEncod
 eBuffer, 39
 Com::Objsys::Asn1::Runtime::Asn1PerOutput
 Stream, 50
 Com::Objsys::Asn1::Runtime::Asn1XerEncod
 eBuffer, 68
 BitCount
 Com::Objsys::Asn1::Runtime::Asn1PerBitFiel
 d, 26
 BitFieldList
 Com::Objsys::Asn1::Runtime::Asn1PerTrace
 Handler, 60
 BitOffset
 Com::Objsys::Asn1::Runtime::Asn1PerBitFiel
 d, 26
 Com::Objsys::Asn1::Runtime::Asn1PerBitFiel
 dList, 27
 Com::Objsys::Asn1::Runtime::Asn1PerDecod
 eBuffer, 35
 Buffer
 Com::Objsys::Asn1::Runtime::Asn1PerEncod
 eBuffer, 44
 Com::Objsys::Asn1::Runtime::Asn1XerEncod
 eBuffer, 72
 ByteAlign
 Com::Objsys::Asn1::Runtime::Asn1PerDecod
 eBuffer, 31
 Com::Objsys::Asn1::Runtime::Asn1PerEncod
 eBuffer, 39
 Com::Objsys::Asn1::Runtime::Asn1PerMessa
 geBuffer, 48
 Com::Objsys::Asn1::Runtime::Asn1PerOutput
 Stream, 50
 ByteArrayInputStream
 Com::Objsys::Asn1::Runtime::Asn1BerEncod
 eBuffer, 11
 Com::Objsys::Asn1::Runtime::Asn1PerEncod
 eBuffer, 44
 ByteIndex
 Com::Objsys::Asn1::Runtime::Asn1PerEncod
 eBuffer, 44
 Bytes
 Com::Objsys::Asn1::Runtime::XmlSource,
 120
 CalcIndefLen
 Com::Objsys::Asn1::Runtime::Asn1BerDecod
 eBuffer, 2
 callBackHandler
 Com::Objsys::Asn1::Runtime::XmlSaxParser,
 116
 Canonical
 Com::Objsys::Asn1::Runtime::Asn1XerEncod
 eBuffer, 72
 Com::Objsys::Asn1::Runtime::Asn1XerOutpu
 tStream, 89
 Characters
 Com::Objsys::Asn1::Runtime::Asn1XerOpen
 Type::SaxHandler, 82
 Com::Objsys::Asn1::Runtime::XmlSaxConten
 tHandler, 103
 Com::Objsys::Asn1::Runtime::XmlSaxDefault
 Handler, 105
 Com::Objsys::Asn1::Runtime::XmlSaxParser
 Adapter, 118
 Com::Objsys::Asn1::Runtime::XmlSource,
 120
 CheckSize
 Com::Objsys::Asn1::Runtime::Asn1BerEncod
 eBuffer, 8
 Com::Objsys::Asn1::Runtime::Asn1PerEncod
 eBuffer, 39

- Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 69
- Clear
 - Com::Objsys::Asn1::Runtime::XmlAttributes, 97
- CloneInstance
 - Com::Objsys::Asn1::Runtime::XmlSaxParser, 114
- Close
 - Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 12
 - Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 24
 - Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 47
 - Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 50
- Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 1
 - Asn1BerDecodeBuffer, 2
 - CalcIndefLen, 2
 - DecodeLength, 2
 - DecodeOpenType, 2
 - DecodeTag, 3
 - DecodeTagAndLength, 3
 - LastTag, 4
 - MatchTag, 3
 - MovePastEOC, 4
 - Parse, 4
 - PeekTag, 4
 - ReadByte, 4
- Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 5
 - Asn1BerDecodeContext, 5
 - Expired, 5
 - MatchElemTag, 5, 6
 - mDecBufByteCount, 6
 - mDecodeBuffer, 6
 - mElemLength, 6
 - mExplicitTagging, 6
 - mTagHolder, 6
- Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 7
 - Asn1BerEncodeBuffer, 7, 8
 - BinDump, 8
 - ByteArrayInputStream, 11
 - CheckSize, 8
 - Copy, 8, 9
 - EncodeIdentifier, 9
 - EncodeIntValue, 9
 - EncodeLength, 9
 - EncodeTag, 9
 - EncodeTagAndLength, 9, 10
 - EncodeUnsignedBinaryNumber, 10
 - GetInputStream, 10
 - MsgCopy, 11
 - MsgLength, 11
 - Reset, 10
 - ToString, 10
 - Write, 10
- Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 11
 - Asn1BerInputStream, 12
 - Available, 12
 - Close, 12
 - Mark, 12
 - MarkSupported, 12
 - Reset, 12
 - Skip, 12
- Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler, 13
 - Asn1BerMessageDumpHandler, 13
 - Contents, 13
 - EndElement, 14
 - StartElement, 14
- Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 14
 - Asn1BerOutputStream, 15
 - Encode, 15
 - EncodeBitString, 15
 - EncodeBMPString, 15
 - EncodeCharString, 16
 - EncodeEOC, 16
 - EncodeIdentifier, 16
 - EncodeIntValue, 16
 - EncodeLength, 16
 - EncodeOctetString, 17
 - EncodeTag, 17
 - EncodeTagAndIndefLen, 17
 - EncodeTagAndLength, 17
 - EncodeUnivString, 18
 - EncodeUnsignedBinaryNumber, 18
- Com::Objsys::Asn1::Runtime::Asn1CerInputStream, 18
 - Asn1CerInputStream, 18
- Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 19
 - Asn1CerOutputStream, 19
 - Encode, 20
 - EncodeBitString, 20
 - EncodeBMPString, 20
 - EncodeCharString, 20
 - EncodeOctetString, 21
 - EncodeStringTag, 21
 - EncodeUnivString, 21
- Com::Objsys::Asn1::Runtime::Asn1DerDecodeBuffer, 22
 - Asn1DerDecodeBuffer, 22
- Com::Objsys::Asn1::Runtime::Asn1DerEncodeBuffer, 22
 - Asn1DerEncodeBuffer, 23

- Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 23
 - Asn1DerInputStream, 23
 - Available, 24
 - Close, 24
 - Mark, 24
 - MarkSupported, 24
 - Reset, 24
 - Skip, 24
- Com::Objsys::Asn1::Runtime::Asn1NotInSetException, 24
 - Asn1NotInSetException, 25
- Com::Objsys::Asn1::Runtime::Asn1PerBitField, 25
 - Asn1PerBitField, 25
 - BitCount, 26
 - BitOffset, 26
 - Name, 26
 - SetBitCountAndOffset, 26
- Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 26
 - AddElemName, 27
 - BitOffset, 27
 - CurrBitField, 28
 - Iterator, 27
 - NewBitField, 27
 - RemoveLastElemName, 27
 - Reset, 27
- Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 28
 - Asn1PerBitFieldPrinter, 28
 - mBitMask, 29
 - mByteIndex, 29
 - mCurrOctet, 29
 - mEncodedMessage, 29
 - mFmtAscCharIdx, 29
 - mFmtBitCharIdx, 29
 - mFmtHexCharIdx, 29
 - mFormatBuffer, 29
 - mPerMessageBuffer, 29
 - Print, 29
- Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 30
 - Asn1PerDecodeBuffer, 31
 - BinDump, 31
 - BitOffset, 35
 - ByteAlign, 31
 - DecodeBit, 31
 - DecodeBitsToInt, 32
 - DecodeBitsToLong, 32
 - DecodeBitsToOctetArray, 32, 33
 - DecodeCharString, 33
 - DecodeConsWholeNumber, 33
 - DecodeExtLength, 33
 - DecodeInt, 34
 - DecodeLength, 34
 - DecodeSmallNonNegWholeNumber, 34
 - IsAligned, 35
 - MoveBitCursor, 35
 - MsgBitCnt, 36
 - mTraceHandler, 36
 - ReadByte, 35
 - SetAligned, 35
 - SetBuffer, 35
 - SetInputStream, 35
 - TraceHandler, 36
- Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler, 36
 - Asn1PerDecodeTraceHandler, 36
 - Enable, 37
 - Print, 37
 - Reset, 37
- Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 37
 - Asn1PerEncodeBuffer, 38
 - BinDump, 39
 - Buffer, 44
 - ByteAlign, 39
 - ByteArrayInputStream, 44
 - ByteIndex, 44
 - CheckSize, 39
 - Copy, 39
 - EncodeBit, 39
 - EncodeBits, 40
 - EncodeCharString, 40
 - EncodeConsWholeNumber, 40
 - EncodeInt, 41
 - EncodeLength, 41, 42
 - EncodeLengthEOM, 42
 - EncodeOctetString, 42
 - EncodeOIDLengthAndValue, 42
 - EncodeOpenType, 42
 - EncodeRelOIDLengthAndValue, 42
 - EncodeSmallNonNegWholeNumber, 43
 - GetInputStream, 43
 - HexDump, 43
 - IsAligned, 43
 - MsgBitCnt, 44
 - MsgByteCnt, 44
 - MsgCopy, 44
 - MsgLength, 44
 - mTraceHandler, 44
 - Reset, 43
 - ReverseBytes, 43
 - SetAligned, 43
 - ToString, 43
 - TraceHandler, 45
 - Write, 44
- Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler, 45
 - Asn1PerEncodeTraceHandler, 45
 - Enable, 45

- Print, 45
- Reset, 46
- Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 46
 - Asn1PerInputStream, 46
 - Available, 46
 - Close, 47
 - Mark, 47
 - MarkSupported, 47
 - Reset, 47
 - Skip, 47
- Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer, 47
 - ByteAlign, 48
 - GetInputStream, 48
 - IsAligned, 48
 - MsgBitCnt, 48
 - TraceHandler, 48
- Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 48
 - AddCaptureBuffer, 50
 - Aligned, 56
 - Asn1PerOutputStream, 50
 - BinDump, 50
 - ByteAlign, 50
 - Close, 50
 - EncodeBit, 51
 - EncodeBits, 51
 - EncodeCharString, 52
 - EncodeConsWholeNumber, 52
 - EncodeInt, 52, 53
 - EncodeLength, 53, 54
 - EncodeLengthEOM, 54
 - EncodeOctetString, 54
 - EncodeOIDLengthAndValue, 54
 - EncodeOpenType, 55
 - EncodeRelOIDLengthAndValue, 55
 - EncodeSmallNonNegWholeNumber, 55
 - Flush, 55
 - mTraceHandler, 56
 - RemoveCaptureBuffer, 55
 - TraceHandler, 57
 - Write, 55, 56
 - WriteByte, 56
- Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler, 57
 - Asn1PerOutputStreamTraceHandler, 57
 - Enable, 57
 - Print, 57
 - Reset, 58
 - ResetTrace, 58
- Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 58
 - AddElemName, 59
 - Asn1PerTraceHandler, 58
 - BitFieldList, 60
 - Enable, 59
 - mBitFieldList, 60
 - NewBitField, 59
 - Print, 59
 - RemoveLastElemName, 59
 - Reset, 59
 - SetBitCount, 59
 - SetBitOffset, 59
- Com::Objsys::Asn1::Runtime::Asn1PerUtil, 60
 - GetMsgBitCnt, 60
- Com::Objsys::Asn1::Runtime::Asn1SetDuplicateException, 60
 - Asn1SetDuplicateException, 61
- Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler, 61
 - Contents, 62
 - EndElement, 62
 - StartElement, 62
- Com::Objsys::Asn1::Runtime::Asn1TagMatchFailedException, 62
 - Asn1TagMatchFailedException, 62, 63
- Com::Objsys::Asn1::Runtime::Asn1XerBase64OctetString, 63
 - Asn1XerBase64OctetString, 63, 64
 - DecodeXER, 64
 - DecodeXML, 64
 - Encode, 64
- Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer, 65
 - Asn1XerDecodeBuffer, 65
 - InputSource, 65
 - mInputSource, 65
- Com::Objsys::Asn1::Runtime::Asn1XerElemInfo, 66
 - Asn1XerElemInfo, 66
 - Equals, 66
 - GetHashCode, 66
 - Optional, 67
- Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 67
 - Asn1XerEncodeBuffer, 68
 - BinDump, 68
 - Buffer, 72
 - Canonical, 72
 - CheckSize, 69
 - Copy, 69
 - DecrLevel, 69
 - EncodeBinStrValue, 69
 - EncodeByte, 69
 - EncodeData, 70
 - EncodeEmptyElement, 70
 - EncodeEndDocument, 70
 - EncodeEndElement, 70
 - EncodeHexStrValue, 70
 - EncodeNamedValue, 70, 71
 - EncodeNamedValueElement, 71

- EncodeObjectId, 71
- EncodeRealValue, 71
- EncodeStartDocument, 71
- EncodeStartElement, 71, 72
- GetInputStream, 72
- IncrLevel, 72
- Indent, 72
- MsgCopy, 73
- MsgLength, 73
- Reset, 72
- State, 73
- Write, 72
- Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 73
 - Copy, 74
 - DecrLevel, 75
 - EncodeBinStrValue, 75
 - EncodeByte, 75
 - EncodeData, 75
 - EncodeEmptyElement, 75, 76
 - EncodeEndDocument, 76
 - EncodeEndElement, 76
 - EncodeHexStrValue, 76
 - EncodeNamedValue, 76, 77
 - EncodeNamedValueElement, 77
 - EncodeObjectId, 77
 - EncodeRealValue, 77
 - EncodeStartDocument, 78
 - EncodeStartElement, 78
 - IncrLevel, 78
 - Indent, 78
 - State, 79
- Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields, 79
 - XERDATA, 79
 - XEREND, 79
 - XERINDENT, 79
 - XERINIT, 79
 - XERSTART, 79
- Com::Objsys::Asn1::Runtime::Asn1XerOpenType, 80
 - Asn1XerOpenType, 80, 81
 - Decode, 81
 - Encode, 81
 - GetSaxHandler, 82
- Com::Objsys::Asn1::Runtime::Asn1XerOpenType::SaxHandler, 82
 - Characters, 82
 - EndElement, 82
 - StartElement, 83
- Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 83
 - Asn1XerOutputStream, 84
 - Canonical, 89
 - Copy, 84, 85
 - DecrLevel, 85
 - EncodeBinStrValue, 85
 - EncodeByte, 85
 - EncodeData, 86
 - EncodeEmptyElement, 86
 - EncodeEndDocument, 86
 - EncodeEndElement, 86
 - EncodeHexStrValue, 86
 - EncodeNamedValue, 87
 - EncodeNamedValueElement, 87
 - EncodeObjectId, 87
 - EncodeRealValue, 87, 88
 - EncodeStartDocument, 88
 - EncodeStartElement, 88
 - IncrLevel, 88
 - Indent, 89
 - State, 89
 - Write, 89
- Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 89
 - Asn1XerSaxHandler, 90
 - Complete, 91
 - Error, 90
 - FatalError, 90
 - Init, 90
 - mCurrElemID, 91
 - mCurrState, 91
 - mLevel, 91
 - mStartLevel, 91
 - State, 92
 - Warning, 91
 - XERDATA, 91
 - XEREND, 91
 - XERINIT, 91
 - XERSTART, 91
 - XERUNKNOWN, 91
- Com::Objsys::Asn1::Runtime::Asn1XerUtil, 92
 - EncodeReal, 92
 - NormalizedRealValueToString, 92
- Com::Objsys::Asn1::Runtime::Asn1XmlEncodeBuffer, 93
 - Asn1XmlEncodeBuffer, 93
 - EncodeEndElement, 93
 - EncodeNamedValueElement, 94
- Com::Objsys::Asn1::Runtime::Asn1XmlOutputStream, 94
 - Asn1XmlOutputStream, 94, 95
 - EncodeEndElement, 95
 - EncodeNamedValueElement, 95
- Com::Objsys::Asn1::Runtime::Asn1XmlUtil, 95
 - EncodeReal, 96
 - GetMinusZero, 96
 - IsMinusZero, 96
- Com::Objsys::Asn1::Runtime::XmlAttributes, 96
 - Add, 97
 - Clear, 97
 - GetFullName, 98

- GetIndex, 98
- GetLength, 98
- GetLocalName, 98
- GetType, 98, 99
- GetURI, 99
- GetValue, 99
- RemoveAttribute, 100
- SetAttribute, 100
- SetAttributes, 100
- SetFullName, 100
- SetLocalName, 100
- SetType, 100
- SetURI, 101
- SetValue, 101
- XmlAttributes, 97
- Com::Objsys::Asn1::Runtime::XmlAttributes::X
mlAttribute, 101
- att_fullName, 102
- att_localName, 102
- att_type, 102
- att_URI, 102
- att_value, 102
- XmlAttribute, 101
- Com::Objsys::Asn1::Runtime::XmlSaxContentH
andler, 102
- Characters, 103
- EndDocument, 103
- EndElement, 103
- EndPrefixMapping, 103
- IgnorableWhitespace, 103
- ProcessingInstruction, 103
- SetDocumentLocator, 103
- SkippedEntity, 104
- StartDocument, 104
- StartElement, 104
- StartPrefixMapping, 104
- Com::Objsys::Asn1::Runtime::XmlSaxDefaultH
andler, 104
- Characters, 105
- EndDocument, 105
- EndElement, 105
- EndPrefixMapping, 105
- Error, 105
- FatalError, 106
- IgnorableWhitespace, 106
- ProcessingInstruction, 106
- ResolveEntity, 106
- SetDocumentLocator, 106
- SkippedEntity, 106
- StartDocument, 106
- StartElement, 107
- StartPrefixMapping, 107
- Warning, 107
- Com::Objsys::Asn1::Runtime::XmlSaxEntityRes
olver, 107
- ResolveEntity, 107
- Com::Objsys::Asn1::Runtime::XmlSaxErrorHan
dler, 108
- Error, 108
- FatalError, 108
- Warning, 108
- Com::Objsys::Asn1::Runtime::XmlSaxLexicalH
andler, 108
- Comment, 109
- EndCDATA, 109
- EndDTD, 109
- EndEntity, 109
- StartCDATA, 109
- StartDTD, 109
- StartEntity, 110
- Com::Objsys::Asn1::Runtime::XmlSaxLocator,
110
- GetColumnNumber, 110
- GetLineNumber, 110
- GetPublicId, 110
- GetSystemId, 110
- Com::Objsys::Asn1::Runtime::XmlSaxLocatorI
mpl, 111
- GetColumnNumber, 111
- GetLineNumber, 112
- GetPublicId, 112
- GetSystemId, 112
- SetColumnNumber, 112
- SetLineNumber, 112
- SetPublicId, 112
- SetSystemId, 112
- XmlSaxLocatorImpl, 111
- Com::Objsys::Asn1::Runtime::XmlSaxParser,
113
- callBackHandler, 116
- CloneInstance, 114
- entityResolver, 116
- errorHandler, 116
- GetContentHandler, 114
- GetEntityResolver, 114
- GetErrorHandler, 114
- lexical, 116
- locator, 116
- namespaceAllowed, 117
- NamespaceAllowed, 117
- NewInstance, 114
- Parse, 114, 115, 116
- parserFileName, 117
- reader, 117
- SetContentHandler, 116
- SetDocumentHandler, 116
- SetEntityResolver, 116
- SetErrorHandler, 116
- XmlSaxParser, 114
- Com::Objsys::Asn1::Runtime::XmlSaxParserAd
apter, 117
- Characters, 118

- EndDocument, 118
- EndElement, 118
- EndPrefixMapping, 118
- IgnorableWhitespace, 118
- ProcessingInstruction, 118
- SetDocumentLocator, 118
- SkippedEntity, 119
- StartDocument, 119
- StartElement, 119
- StartPrefixMapping, 119
- Com::Objsys::Asn1::Runtime::XmlSource, 119
 - Bytes, 120
 - Characters, 120
 - Uri, 120
 - XmlSource, 120
- Comment
 - Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 109
- Complete
 - Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
- Contents
 - Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler, 13
 - Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler, 62
- Copy
 - Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 8, 9
 - Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 39
 - Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 69
 - Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 74
 - Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 84, 85
- CurrBitField
 - Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 28
- Decode
 - Com::Objsys::Asn1::Runtime::Asn1XerOpenType, 81
- DecodeBit
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 31
- DecodeBitsToInt
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 32
- DecodeBitsToLong
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 32
- DecodeBitsToOctetArray
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 32, 33
- DecodeCharString
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 33
- DecodeConsWholeNumber
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 33
- DecodeExtLength
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 33
- DecodeInt
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 34
- DecodeLength
 - Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 2
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 34
- DecodeOpenType
 - Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 2
- DecodeSmallNonNegWholeNumber
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 34
- DecodeTag
 - Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 3
- DecodeTagAndLength
 - Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 3
- DecodeXER
 - Com::Objsys::Asn1::Runtime::Asn1XerBase64OctetString, 64
- DecodeXML
 - Com::Objsys::Asn1::Runtime::Asn1XerBase64OctetString, 64
- DecrLevel
 - Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 69
 - Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 75
 - Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 85
- Enable
 - Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler, 37
 - Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler, 45
 - Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler, 57
 - Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59
- Encode
 - Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 15
 - Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 20

Com::Objsys::Asn1::Runtime::Asn1XerBase64OctetString, 64
Com::Objsys::Asn1::Runtime::Asn1XerOpenType, 81
EncodeBinStrValue
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 69
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 75
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 85
EncodeBit
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 39
Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 51
EncodeBits
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 40
Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 51
EncodeBitString
Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 15
Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 20
EncodeBMPString
Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 15
Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 20
EncodeByte
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 69
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 75
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 85
EncodeCharString
Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 16
Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 20
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 40
Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 52
EncodeConsWholeNumber
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 40
Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 52
EncodeData
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 70
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 75
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 86
EncodeEmptyElement
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 70
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 75, 76
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 86
EncodeEndDocument
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 70
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 76
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 86
EncodeEndElement
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 70
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 76
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 86
Com::Objsys::Asn1::Runtime::Asn1XmlEncodeBuffer, 93
Com::Objsys::Asn1::Runtime::Asn1XmlOutputStream, 95
EncodeEOC
Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 16
EncodeHexStrValue
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 70
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 76
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 86
EncodeIdentifier
Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 9
Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 16
EncodeInt
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 41
Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 52, 53
EncodeIntValue
Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 9
Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 16
EncodeLength

Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 9
 Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 16
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 41, 42
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 53, 54
 EncodeLengthEOM
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 42
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 54
 EncodeNamedValue
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 70, 71
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 76, 77
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 87
 EncodeNamedValueElement
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 71
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 77
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 87
 Com::Objsys::Asn1::Runtime::Asn1XmlEncodeBuffer, 94
 Com::Objsys::Asn1::Runtime::Asn1XmlOutputStream, 95
 EncodeObjectId
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 71
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 77
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 87
 EncodeOctetString
 Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 17
 Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 21
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 42
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 54
 EncodeOIDLengthAndValue
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 42
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 54
 EncodeOpenType
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 42
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 55
 EncodeReal
 Com::Objsys::Asn1::Runtime::Asn1XerUtil, 92
 Com::Objsys::Asn1::Runtime::Asn1XmlUtil, 96
 EncodeRealValue
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 71
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 77
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 87, 88
 EncodeRelOIDLengthAndValue
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 42
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 55
 EncodeSmallNonNegWholeNumber
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 55
 EncodeStartDocument
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 71
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 78
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 88
 EncodeStartElement
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 71, 72
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 78
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 88
 EncodeStringTag
 Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 21
 EncodeTag
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 9
 Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 17
 EncodeTagAndIndefLen
 Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 17
 EncodeTagAndLength
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 9, 10
 Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 17
 EncodeUnivString

Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 18
 Com::Objsys::Asn1::Runtime::Asn1CerOutputStream, 21
 EncodeUnsignedBinaryNumber
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 10
 Com::Objsys::Asn1::Runtime::Asn1BerOutputStream, 18
 EndCDATA
 Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 109
 EndDocument
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 103
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 105
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 118
 EndDTD
 Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 109
 EndElement
 Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler, 14
 Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler, 62
 Com::Objsys::Asn1::Runtime::Asn1XerOpenType::SaxHandler, 82
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 103
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 105
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 118
 EndEntity
 Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 109
 EndPrefixMapping
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 103
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 105
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 118
 entityResolver
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 116
 Equals
 Com::Objsys::Asn1::Runtime::Asn1XerElementInfo, 66
 Error
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 90
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 105
 Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler, 108
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 116
 Expired
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 5
 FatalError
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 90
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
 Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler, 108
 Flush
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 55
 GetColumnNumber
 Com::Objsys::Asn1::Runtime::XmlSaxLocator, 110
 Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 111
 GetContentHandler
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 114
 GetEntityResolver
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 114
 GetErrorHandler
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 114
 GetFullName
 Com::Objsys::Asn1::Runtime::XmlAttributes, 98
 GetHashCode
 Com::Objsys::Asn1::Runtime::Asn1XerElementInfo, 66
 GetIndex
 Com::Objsys::Asn1::Runtime::XmlAttributes, 98
 GetInputStream
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 10
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43
 Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer, 48
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 72
 GetLength
 Com::Objsys::Asn1::Runtime::XmlAttributes, 98
 GetLineNumber
 Com::Objsys::Asn1::Runtime::XmlSaxLocator, 110

Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112

GetLocalName
Com::Objsys::Asn1::Runtime::XmlAttributes, 98

GetMinusZero
Com::Objsys::Asn1::Runtime::Asn1XmlUtil, 96

GetMsgBitCnt
Com::Objsys::Asn1::Runtime::Asn1PerUtil, 60

GetPublicId
Com::Objsys::Asn1::Runtime::XmlSaxLocator, 110
Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112

GetSaxHandler
Com::Objsys::Asn1::Runtime::Asn1XerOpenType, 82

GetSystemId
Com::Objsys::Asn1::Runtime::XmlSaxLocator, 110
Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112

GetType
Com::Objsys::Asn1::Runtime::XmlAttributes, 98, 99

GetURI
Com::Objsys::Asn1::Runtime::XmlAttributes, 99

GetValue
Com::Objsys::Asn1::Runtime::XmlAttributes, 99

HexDump
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43

IgnorableWhitespace
Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 103
Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 118

IncrLevel
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 72
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 78
Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 88

Indent
Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 72
Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 78

Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 89

Init
Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 90

InputSource
Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer, 65

IsAligned
Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 35
Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43
Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer, 48

IsMinusZero
Com::Objsys::Asn1::Runtime::Asn1XmlUtil, 96

Iterator
Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 27

LastTag
Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 4

lexical
Com::Objsys::Asn1::Runtime::XmlSaxParser, 116

locator
Com::Objsys::Asn1::Runtime::XmlSaxParser, 116

Mark
Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 12
Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 24
Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 47

MarkSupported
Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 12
Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 24
Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 47

MatchElemTag
Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 5, 6

MatchTag
Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 3

mBitFieldList
Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 60

mBitMask
Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29

mByteIndex
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mCurrElemID
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 mCurrOctet
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mCurrState
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 mDecBufByteCount
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 6
 mDecodeBuffer
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 6
 mElemLength
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 6
 mEncodedMessage
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mExplicitTagging
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 6
 mFmtAscCharIdx
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mFmtBitCharIdx
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mFmtHexCharIdx
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mFormatBuffer
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 mInputSource
 Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer, 65
 mLevel
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 MoveBitCursor
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 35
 MovePastEOC
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 4
 mPerMessageBuffer
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 MsgBitCnt
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 36
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 44
 Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer, 48
 MsgByteCnt
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 44
 MsgCopy
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 11
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 44
 Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer, 73
 MsgLength
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 11
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 44
 Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer, 73
 mStartLevel
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 mTagHolder
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext, 6
 mTraceHandler
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 36
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 44
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 56
 Name
 Com::Objsys::Asn1::Runtime::Asn1PerBitField, 26
 namespaceAllowed
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 117
 NamespaceAllowed
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 117
 NewBitField
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 27
 Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59
 NewInstance
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 114
 NormalizedRealValueToString
 Com::Objsys::Asn1::Runtime::Asn1XerUtil, 92

Optional
 Com::Objsys::Asn1::Runtime::Asn1XerElemInfo, 67

Parse
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 4
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 114, 115, 116

parserFileName
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 117

PeekTag
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 4

Print
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter, 29
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler, 37
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler, 45
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler, 57
 Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59

ProcessingInstruction
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 103
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 118

ReadByte
 Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer, 4
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 35

reader
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 117

RemoveAttribute
 Com::Objsys::Asn1::Runtime::XmlAttributes, 100

RemoveCaptureBuffer
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 55

RemoveLastElemName
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 27
 Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59

Reset
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 10
 Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 12
 Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 24
 Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList, 27
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler, 37
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler, 46
 Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 47
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler, 58
 Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 72

ResetTrace
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler, 58

ResolveEntity
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
 Com::Objsys::Asn1::Runtime::XmlSaxEntityResolver, 107

ReverseBytes
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43

SetAligned
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 35
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43

SetAttribute
 Com::Objsys::Asn1::Runtime::XmlAttributes, 100

SetAttributes
 Com::Objsys::Asn1::Runtime::XmlAttributes, 100

SetBitCount
 Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59

SetBitCountAndOffset
 Com::Objsys::Asn1::Runtime::Asn1PerBitField, 26

SetBitOffset
 Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler, 59

SetBuffer
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 35

SetColumnNumber
 Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112

SetContentHandler

Com::Objsys::Asn1::Runtime::XmlSaxParser, 116
 SetDocumentHandler
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 116
 SetDocumentLocator
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 103
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 118
 SetEntityResolver
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 116
 SetErrorHandler
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 116
 SetFullName
 Com::Objsys::Asn1::Runtime::XmlAttributes, 100
 SetInputStream
 Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 35
 SetLineNumber
 Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112
 SetLocalName
 Com::Objsys::Asn1::Runtime::XmlAttributes, 100
 SetPublicId
 Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112
 SetSystemId
 Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 112
 SetType
 Com::Objsys::Asn1::Runtime::XmlAttributes, 100
 SetURI
 Com::Objsys::Asn1::Runtime::XmlAttributes, 101
 SetValue
 Com::Objsys::Asn1::Runtime::XmlAttributes, 101
 Skip
 Com::Objsys::Asn1::Runtime::Asn1BerInputStream, 12
 Com::Objsys::Asn1::Runtime::Asn1DerInputStream, 24
 Com::Objsys::Asn1::Runtime::Asn1PerInputStream, 47
 SkippedEntity
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 104
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 119
 StartCDATA
 Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 109
 StartDocument
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 104
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 106
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 119
 StartDTD
 Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 109
 StartElement
 Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler, 14
 Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler, 62
 Com::Objsys::Asn1::Runtime::Asn1XerOpenType::SaxHandler, 83
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 104
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 107
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 119
 StartEntity
 Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler, 110
 StartPrefixMapping
 Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, 104
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 107
 Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter, 119
 State
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 73
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder, 79
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 89
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 92
 ToString
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 10
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 43
 TraceHandler

Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer, 36
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 45
 Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer, 48
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 57
 Uri
 Com::Objsys::Asn1::Runtime::XmlSource, 120
 Warning
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler, 107
 Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler, 108
 Write
 Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer, 10
 Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer, 44
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 55, 56
 Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer, 72
 Com::Objsys::Asn1::Runtime::Asn1XerOutputStream, 89
 WriteByte
 Com::Objsys::Asn1::Runtime::Asn1PerOutputStream, 56
 XERDATA
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields, 79
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 XEREND
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields, 79
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 XERINDENT
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields, 79
 XERINIT
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields, 79
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 XERSTART
 Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields, 79
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 XERUNKNOWN
 Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler, 91
 XmlAttribute
 Com::Objsys::Asn1::Runtime::XmlAttributes:XmlAttribute, 101
 XmlAttributes
 Com::Objsys::Asn1::Runtime::XmlAttributes, 97
 XmlSaxLocatorImpl
 Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl, 111
 XmlSaxParser
 Com::Objsys::Asn1::Runtime::XmlSaxParser, 114
 XmlSource
 Com::Objsys::Asn1::Runtime::XmlSource, 120