ASN1C C# BER/DER/MDER/ OER/PER/XER/XML Runtime

Version 7.8

Objective Systems, Inc.

December 2023

ASN1C C# BER/DER/MDER/OER/PER/XER/XML Runtime

Author's Contact Information. Comments, suggestions, and inquiries regarding ASN1C or this document may be sent by electronic mail to <info@obj-sys.com>.

Table of Contents

List of Tables

Chapter 1. Namespace Documentation

Com

Namespaces

- struct Com::Objsys

Com::Objsys

Namespaces

- struct Com::Objsys::Asn1

Com::Objsys::Asn1

Namespaces

- struct Com::Objsys::Asn1::Runtime

Com::Objsys::Asn1::Runtime

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext
- struct Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1BerInputStream
- struct Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler
- struct Com::Objsys::Asn1::Runtime::Asn1BerOutputStream
- struct Com::Objsys::Asn1::Runtime::Asn1CerInputStream
- struct Com::Objsys::Asn1::Runtime::Asn1CerOutputStream
- struct Com::Objsys::Asn1::Runtime::Asn1DerDecodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1DerEncodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1DerInputStream
- struct Com::Objsys::Asn1::Runtime::Asn1MderDecodeBuffer

  Asn1MderDecodeBuffer provides the source for an MDER decode.

- struct Com::Objsys::Asn1::Runtime::Asn1MderOutputStream
- struct Com::Objsys::Asn1::Runtime::Asn1MderUnsupported

  <summary> Generally, the conditions under which this exception would be thrown should be detected at the time of ASN.1 compilation, so this exception both suggests a code generation error and an incompatibility between the compiled ASN.1 and the MDER.

- struct Com::Objsys::Asn1::Runtime::Asn1NotInSetException
- struct Com::Objsys::Asn1::Runtime::Asn1OerDecodeBuffer

  This class handles the decoding of ASN.1 messages as specified in the Octet Encoding Rules (OER) as documented in the ITU-T X.696 standard.

- struct Com::Objsys::Asn1::Runtime::Asn1OerDecoder
- struct Com::Objsys::Asn1::Runtime::Asn1PerBitField
- struct Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList
- struct Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter
- struct Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler
- struct Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler
- struct Com::Objsys::Asn1::Runtime::Asn1PerInputStream
- struct Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1PerOutputStream
- struct Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler
- struct Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler
- struct Com::Objsys::Asn1::Runtime::Asn1PerUtil
- struct Com::Objsys::Asn1::Runtime::Asn1SetDuplicateException
- struct Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler
- struct Com::Objsys::Asn1::Runtime::Asn1TagMatchFailedException
- struct Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1XerElemInfo
- struct Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1XerEncoder
- struct Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields

- struct Com::Objsys::Asn1::Runtime::Asn1XerOpenType
- struct Com::Objsys::Asn1::Runtime::Asn1XerOutputStream
- struct Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler
- struct Com::Objsys::Asn1::Runtime::Asn1XerUtil
- struct Com::Objsys::Asn1::Runtime::Asn1XmlEncodeBuffer
- struct Com::Objsys::Asn1::Runtime::Asn1XmlOutputStream
- struct Com::Objsys::Asn1::Runtime::Asn1XmlUtil
- struct Com::Objsys::Asn1::Runtime::XmlAttributes
- struct Com::Objsys::Asn1::Runtime::XmlSaxContentHandler
- struct Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler
- struct Com::Objsys::Asn1::Runtime::XmlSaxEntityResolver
- struct Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler
- struct Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler
- struct Com::Objsys::Asn1::Runtime::XmlSaxLocator
- struct Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl
- struct Com::Objsys::Asn1::Runtime::XmlSaxParser
- struct Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter
- struct Com::Objsys::Asn1::Runtime::XmlSource

System

System::Collections

System::Collections::Generic

System::Runtime::InteropServices

Chapter 2. Class Documentation

Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer class Reference

Private Attributes

- Asn1BerExceptionHandler mExcHandler
- Asn1Tag mLastParsedTag
- IntHolder mLenHolder
- System.IO.MemoryStream mOpenTypeCaptureBuffer
- System.IO.MemoryStream mParserCaptureBuffer
- Asn1Tag mTagHolder
-
- static Asn1RunTime rt
- Asn1BerDecodeBuffer (byte [] msgdata)
- Asn1BerDecodeBuffer (System.IO.Stream istream)
- int DecodeEnumValue (bool explicitTagging, int implicitLength)
- int DecodeEnumValue (Asn1Tag tag, bool explicitTagging, int implicitLength)

  <summary> This method decodes an enumerated value from the buffer.
- virtual int DecodeLength ( )
- virtual byte [] DecodeOpenType ( )
- virtual byte [] DecodeOpenType (bool saveData)
- virtual void DecodeTag (Asn1Tag tag)
- virtual int DecodeTagAndLength (Asn1Tag tag)
- System.Exception HandleException (Asn1Exception e)
- virtual bool MatchTag (short tagClass, short tagForm, int tagIDCode, Asn1Tag parsedTag, IntHolder parsedLen)
- virtual bool MatchTag (Asn1Tag tag, Asn1Tag parsedTag, IntHolder parsedLen)
- virtual bool MatchTag (Asn1Tag tag)

- virtual void Parse (Asn1TaggedEventHandler handler)
- virtual void PeekTag (Asn1Tag parsedTag)
- virtual Asn1Tag PeekTag ( )
- override int ReadByte ( )
- void SetExceptionHandler (Asn1BerExceptionHandler handler)
- void SkipTLV ( )
- void ThrowIfUnhandled (Asn1Exception e)
- static int CalcIndefLen (byte [] data, int offset, int len)
- void MovePastEOC (bool saveData)
- void DecodeOpenTypeElement (Asn1Tag tag, IntHolder len, bool saveData)
- void ParseCons (Asn1TaggedEventHandler handler, Asn1Tag tag, int len)
- void ParseElement (Asn1TaggedEventHandler handler, Asn1Tag tag, IntHolder len)
- void ParsePrim (Asn1TaggedEventHandler handler, Asn1Tag tag, int len)

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.

Definition at line 33 of file Asn1BerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeBuffer.cs

Member Data Documentation

Asn1BerExceptionHandler mExcHandler

Exception handler, if one has been set. /p>

Definition at line 41 of file Asn1BerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeBuffer.cs

Asn1BerDecodeBuffer (byte[] msgdata)

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

Table 2.1. 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.

Table 2.2. Parameters

istream	Input stream containing an encoded ASN.1 message.

int DecodeEnumValue (bool explicitTagging, int implicitLength)

This method decodes an enumerated value from the buffer.

Table 2.3. Parameters

explicitTagging	A flag that indicates the element is explicitly tagged.

Table 2.4. Parameters

implicitLength	The length of the contents if implicitly tagged.

Returns: . The decoded integer value.

int DecodeEnumValue (Asn1Tag tag, bool explicitTagging, int implicitLength)

Table 2.5. Parameters

tag	An Asn1Tag value for enumerated values that are tagged other than UNIVERSAL 10.

Table 2.6. Parameters

explicitTagging	A flag that indicates the element is explicitly tagged.

Table 2.7. Parameters

implicitLength	The length of the contents if implicitly tagged.

Returns: . The decoded integer value.

virtual int DecodeLength ()

This method decodes a length value.

Returns: . Decoded length value

virtual byte [] DecodeOpenType ()

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 byte [] DecodeOpenType (bool saveData)

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.

Table 2.8. Parameters

saveData	True if data should be captured and returned

Returns: . Reference to byte array containing component.

virtual void DecodeTag (Asn1Tag tag)

This method decodes a tag value.

Table 2.9. Parameters

tag	Tag object to receive decoded tag fields.

Returns: . status value (see Asn1Status.java)

virtual int DecodeTagAndLength (Asn1Tag tag)

This method decodes a tag and length value.

Table 2.10. Parameters

tag	Tag object to receive decoded tag fields.

Returns: . Decoded length value.

System.Exception HandleException (Asn1Exception e)

If this buffer has an exception handler set, this will return handler.handleException(e, ctxt). Otherwise, this will return e. /p>

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

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 psoitioned at the contents field; otherwise, it will be reset to point to the start of the tag field.

Table 2.11. 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

virtual bool MatchTag (Asn1Tag tag, Asn1Tag parsed-Tag, IntHolder parsedLen)

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

Table 2.12. 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 (Asn1Tag tag)

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

Table 2.13. Parameters

tag	Tag value to be matched.

Returns: . True if given tag matches tag at Decode cursor

virtual void Parse (Asn1TaggedEventHandler handler)

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

Table 2.14. Parameters

handler	Object implementing the Asn1EventHandler interface.

Returns: . Status value

virtual void PeekTag (Asn1Tag parsedTag)

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

Table 2.15. Parameters

parsedTag	Holder object to receive parsed tag value

virtual Asn1Tag PeekTag ()

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

Returns: . Parsed tag object value reference

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

void SetExceptionHandler (Asn1BerExceptionHandler handler)

Set an exception handler on this buffer. /p>

Table 2.16. Parameters

handler	Your exception handler or null.

void SkipTLV ()

This method skips a TLV (tag, length, value) in the encoding. /p>

void ThrowIfUnhandled (Asn1Exception e)

Throw the given exception, unless an exception handler is set and it the handler handles it. /p>

Table 2.17. Parameters

e	The exception<param>

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

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

Table 2.18. Parameters

data	Buffer with the indefinite length message component.
offset	The start offset in the array
len	Length of the buffer (begining from the offset)

Returns: . calculated length

void MovePastEOC (bool saveData)

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.

Table 2.19. Parameters

saveData	True if data should be captured

Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext class Reference

- int mDecBufByteCount
- Asn1BerDecodeBuffer mDecodeBuffer
- int mElemLength

- bool mExplicitTagging
- Asn1Tag mTagHolder

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

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.

Definition at line 33 of file Asn1BerDecodeContext.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeContext.cs

int mDecBufByteCount

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

Definition at line 42 of file Asn1BerDecodeContext.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeContext.cs

Asn1BerDecodeBuffer mDecodeBuffer

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

Definition at line 37 of file Asn1BerDecodeContext.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeContext.cs

int mElemLength

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

Definition at line 47 of file Asn1BerDecodeContext.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeContext.cs

bool mExplicitTagging

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

Definition at line 52 of file Asn1BerDecodeContext.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeContext.cs

Asn1Tag mTagHolder

This variable holds the current parsed tag for matching operations.

Definition at line 55 of file Asn1BerDecodeContext.cs

The Documentation for this struct was generated from the following file:

- Asn1BerDecodeContext.cs

Asn1BerDecodeContext (Asn1BerDecodeBuffer decodeBuffer, int elemLength)

The constructor initializes all internal working varaibles.

Table 2.20. Parameters

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

virtual bool Expired ()

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 (short tagClass, short tagForm, int tagIDCode, IntHolder parsedLen, bool advance)

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.

Table 2.21. 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

virtual bool MatchElemTag (Asn1Tag tag, IntHolder parsedLen, bool advance)

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.

Table 2.22. 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

Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer class Reference

- static Asn1RunTime rt

- Asn1BerEncodeBuffer ( )
- Asn1BerEncodeBuffer (int sizeIncrement)
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- virtual void BinDump ( )
- virtual int EncodeIdentifier (int ident)
- int EncodeIdentifier (BigInteger ident)
- virtual int EncodeIntValue (long ivalue)
- virtual int EncodeLength (int len)
- virtual int EncodeTag (Asn1Tag tag)
- virtual int EncodeTagAndLength (Asn1Tag tag, int len)

- virtual int EncodeTagAndLength (short tagClass, short tagForm, int tagIDCode, int len)
- virtual int EncodeUnsignedBinaryNumber (long ivalue)
- virtual bool IsBER ( )
- virtual int TrimBitString (Asn1BitString bitstr)
- override void CheckSize (int bytesRequired)

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.

Definition at line 36 of file Asn1BerEncodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1BerEncodeBuffer.cs

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.

Table 2.23. Parameters

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

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.

Table 2.24. Parameters

outs	StreamWriter where dump will be printed

varName	Name of the Decoded ASN1 Type

virtual void BinDump ()

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

virtual int EncodeIdentifier (int ident)

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

Table 2.25. Parameters

ident	The identifier to be encoded.

Returns: . Length of the encoded component in octets.

int EncodeIdentifier (BigInteger ident)

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

Table 2.26. Parameters

ident	The identifier to be encoded.

Returns: . Length of the encoded component in octets.

virtual int EncodeIntValue (long ivalue)

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

Table 2.27. Parameters

ivalue	Integer value to encode

Returns: . Length of encoded component

virtual int EncodeLength (int len)

This method encodes a length value.

Table 2.28. Parameters

len	The length to be encoded.

Returns: . Length of encoded component

virtual int EncodeTag (Asn1Tag tag)

This method encodes a tag value.

Table 2.29. Parameters

tag	The tag to be encoded.

Returns: . Length of component or negative status value

virtual int EncodeTagAndLength (Asn1Tag tag, int len)

This method encodes both a tag and length value.

Table 2.30. Parameters

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

Returns: . Length of encoded component

virtual int EncodeTagAndLength (short tagClass, short tagForm, int tagIDCode, int len)

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

Table 2.31. 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 EncodeUnsignedBinaryNumber (long ivalue)

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

Table 2.32. Parameters

ivalue	Integer value to encode

Returns: . Length of encoded component

virtual int TrimBitString (Asn1BitString bitstr)

This method will trim a BIT STRING for DER encoding by removing all zero trailing bits. The default implementation in Asn1BerEncodeBuffer does nothing. The overriden version in Asn1DerEncodeBuffer will trim the string.

<param name="bitstr> ASN.1 BIT STRING object <return> Adjusted bit count </return>

override void CheckSize (int bytesRequired)

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

Table 2.33. Parameters

bytesRequired	Number of required bytes.

Com::Objsys::Asn1::Runtime::Asn1BerInputStream class Reference

- 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)

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.

Definition at line 33 of file Asn1BerInputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1BerInputStream.cs

Asn1BerInputStream (System.IO.Stream istream)

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

Table 2.34. Parameters

istream	Input stream containing an encoded ASN.1 message.

virtual int Available ()

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

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

Table 2.35. Exceptions

System.SystemException	if an I/O error occurs.

virtual void Close ()

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

Table 2.36. 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 reseting the input stream position to current position.

virtual bool MarkSupported ()

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

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

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.

Table 2.37. Parameters

nbytes	Number of bytes to skip

Table 2.38. Exceptions

System.SystemException	an I/O error occurs.

Returns: . Skipped number of bytes

Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpH class Reference

- const int MaxBytesPerLine
- int mOffset
- System.IO.StreamWriter mPrintStream

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

- void PrintOffset ( )

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.

Definition at line 38 of file Asn1BerMessageDumpHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1BerMessageDumpHandler.cs

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.

Table 2.39. Parameters

outs	Output stream for formatted data

virtual void Contents (byte[] data)

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

Table 2.40. Parameters

data	Array containing the encoded contents bytes

virtual void EndElement (Asn1Tag tag)

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

Table 2.41. Parameters

tag	Array containing the encoded contents bytes

virtual void StartElement (Asn1Tag tag, int len, byte[] tagLenBytes)

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

Table 2.42. Parameters

tag	Parsed tag value
len	Parsed length value
tagLenBytes	Array containing the encoded tag/length bytes

Com::Objsys::Asn1::Runtime::Asn1BerOutputStream class Reference

- static readonly byte [] EOC

- static Asn1RunTime rt

- Asn1BerOutputStream (System.IO.Stream os)
- Asn1BerOutputStream (System.IO.Stream os, int bufSize)
- 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 (Asn1Tag tag)
- virtual void EncodeTag (short tagClass, short tagForm, int tagIDCode)
- virtual void EncodeTagAndIndefLen (Asn1Tag tag)
- virtual void EncodeTagAndIndefLen (short tagClass, short tagForm, int tagIDCode)
- virtual void EncodeTagAndLength (Asn1Tag tag, int len)
- virtual void EncodeUnivString (int [] data, bool explicitTagging, Asn1Tag tag)
- virtual void EncodeUnsignedBinaryNumber (long data)
- virtual bool IsBER ( )

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.

Definition at line 38 of file Asn1BerOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1BerOutputStream.cs

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.

Table 2.43. 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.

Table 2.44. 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.

virtual void Encode (Asn1Type type, bool explicitTagging)

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.

Table 2.45. Parameters

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

virtual void EncodeBitString (byte[] data, int numbits, bool explicitTagging, Asn1Tag tag)

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

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

Table 2.46. 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

Table 2.47. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeBMPString (System.String data, bool explicitTagging, Asn1Tag tag)

This method writes the given string as BMP string value.

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

Table 2.48. Parameters

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

Table 2.49. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeCharString (System.String data, bool explicitTagging, Asn1Tag tag)

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.

Table 2.50. Parameters

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

Table 2.51. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeEOC ()

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)

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.

Table 2.52. Parameters

ident	The identifier to be encoded.

virtual void EncodeIntValue (long data, bool encodeLen)

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.

Table 2.53. Parameters

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

virtual void EncodeLength (int len)

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

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

Table 2.54. Parameters

len	The length to be encoded.

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

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

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

Table 2.55. Parameters

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

Table 2.56. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeTag (Asn1Tag tag)

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

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

Table 2.57. Parameters

tag	The tag to be encoded.

virtual void EncodeTag (short tagClass, short tagForm, int tagIDCode)

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

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

Table 2.58. 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)

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.

Table 2.59. Parameters

tag	The tag to be encoded.

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

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.

Table 2.60. 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 EncodeTagAndLength (Asn1Tag tag, int len)

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

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

Table 2.61. Parameters

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

virtual void EncodeUnivString (int[] data, bool explicit-Tagging, Asn1Tag tag)

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

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

Table 2.62. Parameters

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

Table 2.63. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeUnsignedBinaryNumber (long data)

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.

Table 2.64. Parameters

data	Integer value to encode.

Com::Objsys::Asn1::Runtime::Asn1CerInputStream class Reference

- Asn1CerInputStream (System.IO.Stream istream)

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.

Definition at line 33 of file Asn1CerInputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1CerInputStream.cs

Asn1CerInputStream (System.IO.Stream istream)

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

Table 2.65. Parameters

istream	Input stream containing an encoded ASN.1 message.

Com::Objsys::Asn1::Runtime::Asn1CerOutputStream class Reference

- static Asn1RunTime rt

- Asn1CerOutputStream (System.IO.Stream os)
- Asn1CerOutputStream (System.IO.Stream os, int bufSize)

- 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, Asn1Tag tag)
- virtual void EncodeStringTag (int nbytes, short tagClass, short tagForm, int tagIDCode)
- override void EncodeUnivString (int [] value, bool explicitTagging, Asn1Tag tag)
- override bool IsBER ( )

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.

Definition at line 38 of file Asn1CerOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1CerOutputStream.cs

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.

Table 2.66. 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.

Table 2.67. 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.

override void Encode (Asn1Type type, bool explicitTagging)

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.

Table 2.68. Parameters

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

override void EncodeBitString (byte[] value, int numbits, bool explicitTagging, Asn1Tag tag)

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

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

Table 2.69. 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

Table 2.70. Exceptions

Asn1Exception	Thrown, if operation is failed.

override void EncodeBMPString (System.String value, bool explicitTagging, Asn1Tag tag)

This method writes the given string as BMP string value.

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

Table 2.71. Parameters

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

tag	Universal tag to apply

Table 2.72. Exceptions

Asn1Exception	Thrown, if operation is failed.

override void EncodeCharString (System.String value, bool explicitTagging, Asn1Tag tag)

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.

Table 2.73. Parameters

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

Table 2.74. Exceptions

Asn1Exception	Thrown, if operation is failed.

override void EncodeOctetString (byte[] value, bool explicitTagging, Asn1Tag tag)

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

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

Table 2.75. Parameters

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

Table 2.76. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeStringTag (int nbytes, Asn1Tag tag)

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

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

Table 2.77. Parameters

nbytes	The number of bytes in string to be encoded.
tag	The tag to be encoded.

virtual void EncodeStringTag (int nbytes, short tagClass, short tagForm, int tagIDCode)

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

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

Table 2.78. 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.

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

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

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

Table 2.79. Parameters

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

Table 2.80. Exceptions

Asn1Exception	Thrown, if operation is failed.

Asn1DecodeBitBuffer class Reference

Asn1DecodeBuffer class Reference

Com::Objsys::Asn1::Runtime::Asn1DerDecodeBuffer class Reference

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

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.

Definition at line 33 of file Asn1DerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1DerDecodeBuffer.cs

Asn1DerDecodeBuffer (byte[] msgdata)

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

Table 2.81. 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.

Table 2.82. Parameters

istream	Input stream containing an encoded ASN.1 message.

Com::Objsys::Asn1::Runtime::Asn1DerEncodeBuffer class Reference

- Asn1DerEncodeBuffer ( )

- Asn1DerEncodeBuffer (int sizeIncrement)
- override bool IsBER ( )
- override int TrimBitString (Asn1BitString bitstr)

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.

Definition at line 34 of file Asn1DerEncodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1DerEncodeBuffer.cs

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.

Table 2.83. Parameters

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

override int TrimBitString (Asn1BitString bitstr)

This method will trim a BIT STRING for DER encoding by removing all zero trailing bits.

<param name="bitstr> ASN.1 BIT STRING object <return> Adjusted bit count </return>

Com::Objsys::Asn1::Runtime::Asn1DerInputStream class Reference

- 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)

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.

Definition at line 33 of file Asn1DerInputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1DerInputStream.cs

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.

Table 2.84. Parameters

istream	Input stream containing an encoded ASN.1 message.

virtual int Available ()

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

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 ()

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 reseting the input stream position to current position.

virtual bool MarkSupported ()

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

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

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.

Table 2.85. Parameters

nbytes	Number of bytes to skip

Returns: . Skipped number of bytes

Asn1EncodeBitBuffer class Reference

Asn1EncodeByteBuffer class Reference

Asn1EncodeByteBufferRev class Reference

Asn1Exception class Reference

Asn1InputStream class Reference

Com::Objsys::Asn1::Runtime::Asn1MderDecodeBuffer class Reference

- Asn1MderDecodeBuffer (byte [] msgdata)

  Convenience method to create a decoder on an array of data.

- Asn1MderDecodeBuffer (System.IO.Stream istream)

  Create a decoder on the given input stream.

- int Available ( )
- void Close ( )
- bool MarkSupported ( )
- override int ReadByte ( )

Asn1MderDecodeBuffer (byte[] msgdata)

Table 2.86. Parameters

msgdata	

Asn1MderDecodeBuffer (System.IO.Stream istream)

Table 2.87. Parameters

istream	

Com::Objsys::Asn1::Runtime::Asn1MderOutputStream class Reference

- Asn1MderOutputStream (System.IO.Stream os)

Detailed Description

An output stream for use with MDER encoding. /p> To the normal output stream functionality, this class adds some methods that are useful to doing MDER encoding.

Definition at line 32 of file Asn1MderOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1MderOutputStream.cs

Com::Objsys::Asn1::Runtime::Asn1MderUnsupported class Reference

- Asn1MderUnsupported (System.String msg)

Detailed Description

The ASN1C runtime throws this exception whenever it is unable to proceed due to the restrictive nature of the MDER encoding rules.

Definition at line 39 of file Asn1MderUnsupported.cs

The Documentation for this struct was generated from the following file:

- Asn1MderUnsupported.cs

Com::Objsys::Asn1::Runtime::Asn1NotInSetException class Reference

- Asn1NotInSetException (Asn1BerDecodeBuffer buffer, Asn1Tag tag)

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.

Definition at line 36 of file Asn1NotInSetException.cs

The Documentation for this struct was generated from the following file:

- Asn1NotInSetException.cs

Asn1NotInSetException (Asn1BerDecodeBuffer buffer, Asn1Tag tag)

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

Table 2.88. Parameters

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

Com::Objsys::Asn1::Runtime::Asn1OerDecodeBuffer class Reference

Private Attributes

- bool mbCanonical

- static Asn1RunTime rt

- Asn1OerDecodeBuffer (byte [] msgdata)

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

- Asn1OerDecodeBuffer (System.IO.Stream istream)

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

- int DecodeEnumValue ( )

Decode enumerated value encoded in OER.

- long DecodeIntSigned (int octets)

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

- long DecodeIntSigned ( )

  Decode an integer value as a variable length, signed integer, including decoding the length, according to OER.

- long DecodeIntUnsigned (int octets)
- long DecodeIntUnsigned ( )
- int DecodeLength ( )

  Decode OER length determinant and return the decoded value.

- int DecodeQuantity ( )

  Decode an OER quantity (used for SEQUENCE-OF and SET-OF)

- void DecodeTag (Asn1Tag tag)

  Decode a tag encoded in OER.

- long DecodeVarUnsigned (int octets)
- bool GetCanonicalMode ( )
- void SetCanonicalMode (bool value)

- void checkCanonicalSigned (long value, int octets)
- long DecodeVarSigned (int numOcts)

Asn1OerDecodeBuffer (byte[] msgdata)

Table 2.89. Parameters

msgdata	Byte array containing an encoded ASN.1 message.

Asn1OerDecodeBuffer (System.IO.Stream istream)

In this case, the message is passed in using an InputStream object.

Table 2.90. Parameters

istream	Input stream containing an encoded ASN.1 message.

long DecodeIntSigned (int octets)

Note: this function will do sign-extension so that if a negative value was encoded in less than 8 bytes, the returned long will be that value (and not some positive value).

Table 2.91. Parameters

octets;	0 < octets <= 8

Returns: .

long DecodeIntSigned ()

This is used for integer values that have a lower bound less than -2^63, no lower bound, or a lower bound less than zero in combination with an upper bound greater than 2^63-1, or no upper bound. (In other words, it doesn't fit in a signed 64-bit integer.)

long DecodeIntUnsigned (int octets)

Decode an unsigned integer value (a binary integer, not 2's complement form), of the given length. /p>

Table 2.92. Parameters

octets	The number of octets; 0 < octets <= 8

long DecodeIntUnsigned ()

Decode an integer value as a variable length, unsigned integer, including decoding the length, according to OER. This is used for integer values that are constrained to be non-negative but which have no upper bound or an upper bound greater than 2^64 - 1.

int DecodeQuantity ()

Returns: .

Table 2.93. Exceptions

IOException	

void DecodeTag (Asn1Tag tag)

Table 2.94. Parameters

tag	Object to decode into.

long DecodeVarUnsigned (int octets)

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

This will throw an exception if canonical mode is set and the encoding used extra bytes.

Note: if octets >= 8, this method will throw an exception if the encoded integer is outside the range of a long; the exception is not necessarily thrown as the integer might not have been encoded in the minimal number of octets.

Table 2.95. Parameters

octets	The number of octets; 0 < octets

bool GetCanonicalMode ()

Return true if canonical mode has been indicated by calling SetCanonicalMode(true);

void SetCanonicalMode (bool value)

Turn canonical mode on/off. Turning canonical mode on acts as a signal to both generated code and runtime code that the user wants to enforce the canonical OER rules. It is not required to turn on canonical mode just because the encoding is canonical.

void checkCanonicalSigned (long value, int octets)

Throw an exception if the given long value, which is assumed to be representable in the given number of octets, can be represented as a signed number in still fewer octets than the given number of octets.

long DecodeVarSigned (int numOcts)

Decode an OER variable-sized signed number.

This will throw an exception if canonical mode is set and the encoding used extra bytes.

Table 2.96. Parameters

numOcts	The number of octets to decode. The decoded octets will be in the lowest octets of the returned value.

Com::Objsys::Asn1::Runtime::Asn1OerDecoder interface Reference

- Asn1Type Decode (Asn1OerDecodeBuffer buffer)

  <summmary> Decode value from given buffer.

Detailed Description

This interface provides an interface for decoding. It is meant to be implemented by generated classes corresponding to enumerated types. It is useful for cases where a non-static decode method is needed, such as during decoding driven by table constraints, where the type to be decoded is not known at compile time.

Definition at line 36 of file Asn1OerDecoder.cs

The Documentation for this struct was generated from the following file:

- Asn1OerDecoder.cs

Asn1Type Decode (Asn1OerDecodeBuffer buffer)

</summmary>

Asn1OpenType class Reference

Asn1OutputStream class Reference

Com::Objsys::Asn1::Runtime::Asn1PerBitField class Reference

Private Attributes

- int mBitCount
- int mBitOffset
- System.String mName
-
-
-
- Asn1PerBitField (System.String name, int bitOffset, int bitCount)
- virtual void SetBitCountAndOffset (int count, int offset)

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.

Definition at line 35 of file Asn1PerBitField.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitField.cs

Asn1PerBitField (System.String name, int bitOffset, int bitCount)

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

Table 2.97. Parameters

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

virtual void SetBitCountAndOffset (int count, int offset)

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

Table 2.98. Parameters

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

Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList class Reference

Private Attributes

- System.Collections.ArrayList mBitFieldList
- Asn1PerBitField mCurrBitField
- int mLastElemNameStartIndex
- System.Text.StringBuilder mNameBuffer
-
-
- 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 ( )
- void ReplaceLastFieldWithDetail (Asn1PerBitFieldList details)
- virtual void Reset ( )

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.

Definition at line 34 of file Asn1PerBitFieldList.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldList.cs

virtual void AddElemName (System.String name, int arrayx)

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).

Table 2.99. 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 ()

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)

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.

Table 2.100. 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 ()

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.

void ReplaceLastFieldWithDetail (Asn1PerBitFieldList details)

Replaces the last bit field in this bit trace handler with the fields from the given trace handler, which are assumed to be a breakdown of the last field.

Table 2.101. Parameters

details	field list to take details from

virtual void Reset ()

This method resets the object.

Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter class Reference

Private Attributes

- const int ASCCHAROFFSET
- const int HEXCHAROFFSET
- int mFmtAscCharIdx
- int mFmtHexCharIdx
- int mBitMask
- int mByteIndex
- int mCurrOctet
- System.IO.Stream mEncodedMessage
- int mFmtBitCharIdx
- System.Text.StringBuilder mFormatBuffer
- Asn1PerMessageBuffer mPerMessageBuffer
- Asn1PerBitFieldPrinter (Asn1PerMessageBuffer perMessageBuffer, System.IO.Stream encodedMessage)

- virtual void Print (System.IO.StreamWriter outs, System.String varName)

- void FmtAndPrintBit (System.IO.StreamWriter outs, int data)
- void Print (System.IO.StreamWriter outs, System.String varName, Asn1PerBitField bitField, int nextBitOffset)
- void ResetFormatBuffer ( )

Detailed Description

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

Definition at line 34 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

int mBitMask

This variable holds the mask for current bit

Definition at line 39 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

int mByteIndex

This variable holds the byte index

Definition at line 41 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

int mCurrOctet

This variable holds the current byte

Definition at line 43 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

System.IO.Stream mEncodedMessage

This variable holds the input stream

Definition at line 50 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

int mFmtBitCharIdx

This variable holds the index for formated information

Definition at line 48 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

System.Text.StringBuilder mFormatBuffer

This variable holds the formated information of current byte

Definition at line 45 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

Asn1PerMessageBuffer mPerMessageBuffer

This variable holds the PER encode or decode message buffer

Definition at line 52 of file Asn1PerBitFieldPrinter.cs

The Documentation for this struct was generated from the following file:

- Asn1PerBitFieldPrinter.cs

Asn1PerBitFieldPrinter (Asn1PerMessageBuffer perMessageBuffer, System.IO.Stream encodedMessage)

Constructor

Table 2.102. Parameters

perMessageBuffer	PER encode or decode message buffer
encodedMessage	Input stream of encoded message

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

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.

Table 2.103. Parameters

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

Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer class Reference

Private Attributes

- bool mbAligned
- Asn1SizeConstraint mSizeConstraint
-
- Asn1PerTraceHandler mTraceHandler
- static Asn1RunTime rt
- void InitBlock (bool aligned)
- Asn1PerDecodeBuffer (byte [] msgdata, bool aligned)
- Asn1PerDecodeBuffer (System.IO.Stream istream, bool aligned)
- virtual void BinDump (System.String varName)
- virtual void BinDump (System.IO.StreamWriter outs, System.String varName)
- override void ByteAlign ( )
- virtual bool DecodeBit (System.String ident)
- override bool DecodeBit ( )
- virtual int DecodeBitsToInt (int nbits, System.String ident)
- override int DecodeBitsToInt (int nbits)
- virtual long DecodeBitsToLong (int nbits, System.String ident)
- override long DecodeBitsToLong (int nbits)
- virtual void DecodeBitsToOctetArray (byte [] data, int offset, int nbits, System.String ident)
- virtual void DecodeBitsToOctetArray (byte [] data, int offset, int bitOffset, int nbits, System.String ident)
- override void DecodeBitsToOctetArray (byte [] data, int offset, int nbits)
- virtual void DecodeCharString (int nchars, int abpc, int ubpc, Asn1CharSet charSet, System.Text.StringBuilder sbuf)

- virtual long DecodeConsWholeNumber (long rangeValue, System.String ident)
- virtual long DecodeConsWholeNumber (long rangeValue, bool retval, System.String ident)
- virtual long DecodeConsWholeNumber (long rangeValue)
- virtual long DecodeExtLength ( )
- virtual long DecodeInt (int nocts, bool signExtend, System.String ident)
- virtual long DecodeInt (int nocts, bool signExtend)
- virtual long DecodeLength ( )
- virtual long DecodeLength (long lower, long upper)
- virtual int DecodeSmallLength ( )
- virtual int DecodeSmallNonNegWholeNumber ( )
- virtual long DecodeUnconsLength ( )
- virtual bool IsAligned ( )
- virtual void SetAligned (bool data)
- void SetSizeConstraint (long lower, long upper)
- void SetSizeConstraintExt (long lower, long upper, long extLower, long extUpper)
- static Asn1PerDecodeBuffer SetBuffer (Asn1PerDecodeBuffer buffer, byte [] msgdata, bool aligned)

Detailed Description

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

Definition at line 33 of file Asn1PerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1PerDecodeBuffer.cs

Asn1PerTraceHandler mTraceHandler

Variable holds the PER message trace handler

Definition at line 885 of file Asn1PerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1PerDecodeBuffer.cs

Asn1PerDecodeBuffer (byte[] msgdata, bool aligned)

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

Table 2.104. 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.

Table 2.105. Parameters

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

virtual void BinDump (System.String varName)

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

Table 2.106. Parameters

varName	Name of top-level message object variable

virtual 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 output stream.

Table 2.107. Parameters

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

override void ByteAlign ()

This methods byte-aligns the buffer. If the buffer was created unaligned, this does nothing.

virtual bool DecodeBit (System.String ident)

This method decodes a single bit value.

Table 2.108. Parameters

ident	Bit field identifier name for tracing.

Returns: . Boolean value of bit that was decoded.

override bool DecodeBit ()

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 int DecodeBitsToInt (int nbits, System.String ident)

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.

Table 2.109. Parameters

nbits	Number of bits to Decode
ident	Bit field identifier name for tracing.

Returns: . Integer value containing decoded bits

override int DecodeBitsToInt (int nbits)

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

Table 2.110. Parameters

nbits	Number of bits to Decode

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

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

Table 2.111. Parameters

nbits	Number of bits to Decode
ident	Bit field identifier name for tracing.

override long DecodeBitsToLong (int nbits)

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'.

Table 2.112. Parameters

nbits	Number of bits to Decode

Returns: . Long integer value containing decoded bits

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

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.

Table 2.113. 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 DecodeBitsToOctetArray (byte[] data, int offset, int bitOffset, int nbits, System.String ident)

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 first bit is decoded into the given offset byte at the MSB if bitOffset == 0, and at the LSB if bitOffset == 7.

Table 2.114. Parameters

data	Octet array for decoded data
offset	Starting byte offset into array
bitOffset	Where in first byte the first bit goes

nbits	Number of bits to Decode
ident	Bit field identifier name for tracing.

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

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'.

Table 2.115. Parameters

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

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

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

Table 2.116. 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	It not null, string buffer to receive decoded result

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

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

Table 2.117. Parameters

rangeValue	upper - lower + 1. Treated as unsigned, with 0 representing 2^64.

ident	Tracing identifier

Returns: . Decoded adjusted value = value - lower range endpoint value

virtual long DecodeConsWholeNumber (long rangeValue, bool retval, System.String ident)

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

Table 2.118. Parameters

rangeValue	upper - lower + 1. Treated as unsigned, with 0 representing 2^64.
ident	Tracing identifier
retval	If true, return decoded adjusted value; else return 0.

Returns: . Decoded adjusted value = value - lower range endpoint value

virtual long DecodeConsWholeNumber (long rangeValue)

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'.

Table 2.119. Parameters

rangeValue	lower - upper + 1

Returns: . Decoded adjusted value = value - lower range endpoint value

virtual long DecodeExtLength ()

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, System.String ident)

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

Returns: . Decoded long integer value

Table 2.120. Parameters

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

virtual long DecodeInt (int nocts, bool signExtend)

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

Table 2.121. Parameters

nocts	Number of octets to Decode
signExtend	Sign extend resulting value

virtual long DecodeLength ()

This method decodes either a constrained or unconstrained length depending on whether a size constraint object is set in this object.

Returns: . Decoded length value.

virtual long DecodeLength (long lower, long upper)

This method decodes a constrained length determinant value.

Table 2.122. Parameters

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

Returns: . Decoded length value

virtual int DecodeSmallLength ()

This method implements the rules to Decode a normally small length as specified in section 11.9 of the X.691 standard.

Returns: . Decoded int value

virtual int DecodeSmallNonNegWholeNumber ()

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 long DecodeUnconsLength ()

This method decodes a general (unconstrained) length determinant value as described in section 11.9 of the 2008 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 >= 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 bool IsAligned ()

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

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

virtual void SetAligned (bool data)

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

Table 2.123. Parameters

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

void SetSizeConstraint (long lower, long upper)

This method is used to set a size constraint within the buffer object. The constraint will only be set if an existing constraint is not already in place (i.e. if the existing reference is not null). This is used for length decoding of SEQUENCE OF objects to pass constraints applied to referenced objects to the base object.

void SetSizeConstraintExt (long lower, long upper, long extLower, long extUpper)

This method is used to set an extensible size constraint in the buffer object. The constraint will only be set if an existing constraint is not already in place (i.e. if the existing reference is not null). This is used for length encoding of SEQUENCE OF objects to pass constraints applied to referenced objects to the base object.

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

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.

Table 2.124. 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

Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHa class Reference

Private Attributes

- System.IO.MemoryStream mCaptureBuffer
- Asn1PerDecodeBuffer mMessageBuffer

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

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.

Definition at line 38 of file Asn1PerDecodeTraceHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1PerDecodeTraceHandler.cs

Asn1PerDecodeTraceHandler (Asn1PerDecodeBuffer messageBuffer)

This constructor initializes the internal trace handler member variables.

Table 2.125. Parameters

messageBuffer	PER decode message buffer object reference

override void Enable ()

This method is used to turn PER bit tracing on

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

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

Table 2.126. Parameters

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

override void Reset ()

This method resets the trace bit field list.

Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer class Reference

Private Attributes

- bool mbAligned
- Asn1SizeConstraint mSizeConstraint
-
- static Asn1RunTime rt
- static readonly byte [] ZEROBYTE
- Asn1PerTraceHandler mTraceHandler
- void EncodeIdentifier (long ident)
- void InitBlock (bool aligned)
- Asn1PerEncodeBuffer (bool aligned)
- Asn1PerEncodeBuffer (bool aligned, int sizeIncrement)
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- override void ByteAlign ( )
- virtual void EncodeBit (bool value, System.String ident)
- override void EncodeBit (bool value)

- virtual void EncodeBits (byte [] value, int offset, int nbits, System.String ident)
- virtual void EncodeBits (byte [] value, int offset, int bitOffset, int nbits, System.String ident)
- override void EncodeBits (byte [] value, int offset, int bitOffset, int nbits)
- override void EncodeBits (byte [] value, int offset, 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, System.String ident)
- virtual void EncodeConsWholeNumber (long adjustedValue, long rangeValue)
- virtual void EncodeInt (long value, int nbits, System.String ident)
- virtual void EncodeInt (long value, int nbits)
- virtual void EncodeInt (long value, bool encodeLen, bool signExtend, System.String ident)
- virtual void EncodeInt (long value, bool encodeLen, bool signExtend)
- virtual long EncodeLength (long value)
- virtual void EncodeLength (long value, long lower, long upper)
- 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 EncodeOpenType (Asn1PerEncodeBuffer buffer, System.String elemName)
- virtual void EncodeRelOIDLengthAndValue (int [] value)
- virtual void EncodeSmallLength (int value)
- virtual void EncodeSmallNonNegWholeNumber (int value)
- virtual long EncodeUnconsLength (long value)
- override void HexDump ( )
- virtual bool IsAligned ( )
- override void Reset ( )
- virtual void SetAligned (bool value)
- void SetSizeConstraint (long lower, long upper)
- void SetSizeConstraintExt (long lower, long upper, long extLower, long extUpper)

Detailed Description

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

Definition at line 38 of file Asn1PerEncodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1PerEncodeBuffer.cs

Asn1PerTraceHandler mTraceHandler

Variable holds the PER message trace handler

Definition at line 1071 of file Asn1PerEncodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1PerEncodeBuffer.cs

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.

Table 2.127. 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.

Table 2.128. 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.

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.

Table 2.129. Parameters

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

override void ByteAlign ()

This method byte-aligns the buffer if the buffer is set to be aligned.

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

This method encodes a single bit value.

Table 2.130. Parameters

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

override void EncodeBit (bool value)

This method encodes a single bit value. The `ident` argument which is used for tracing is defaulted to 'value'. Equivalent to EncodeBit(value, "value");

Table 2.131. Parameters

value	Boolean value of bit to be encoded.

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

This method encodes bit values from an array of octets.

Table 2.132. 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 offset, int bitOffset, int nbits, System.String ident)

This method encodes bit values from an array of octets.

Table 2.133. Parameters

value	Octet array containing bits to be encoded

offset	Starting byte offset in value
bitOffset	Starting bit offset in first byte. For example, passing 0 would begin encoding with the most significant bit, while passing 7 would begin encoding with the least significant bit.
nbits	Number of bits to encode
ident	Bit field identifier name for tracing.

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

This method encodes bit values from an array of octets.

This overrides the parent class method in order to default the name for tracing to "value". Equivalent to: encodeBits(value, offset, bitOffset, nbits, "value");

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

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

Table 2.134. Parameters

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

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

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

Table 2.135. 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 adjusted-Value, long rangeValue, System.String ident)

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

Table 2.136. Parameters

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

virtual void EncodeConsWholeNumber (long adjusted-Value, long rangeValue)

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'.

Table 2.137. Parameters

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

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

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

Table 2.138. Parameters

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

virtual void EncodeInt (long value, int nbits)

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'.

Table 2.139. Parameters

value	Integer containing bits to be encoded

nbits	Number of bits to encode

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

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.

Table 2.140. 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, bool encodeLen, bool signExtend)

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'.

Table 2.141. 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 long EncodeLength (long value)

This method encodes either a constrained or unconstrained length depending on whether a size constraint object is set in this object.

Table 2.142. 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 fragementation was done (i.e the value was >= 16k).

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

This method encodes a constrained length determinant value.

Table 2.143. Parameters

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

virtual void EncodeLengthEOM (long value)

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.

Table 2.144. Parameters

value	Original length value that was encoded.

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

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

Table 2.145. 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)

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

Table 2.146. Parameters

value	Integer array containing arcs to encode

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

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

Table 2.147. 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 EncodeOpenType (Asn1PerEncodeBuffer buffer, System.String elemName)

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

Table 2.148. Parameters

buffer	PER encode buffer containing encoded message component.
elemName	Name of element being encoded.

virtual void EncodeRelOIDLengthAndValue (int[] value)

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

Table 2.149. Parameters

value	Integer array containing arcs to encode

virtual void EncodeSmallLength (int value)

This method implements the rules to encode a normally small length as specified in section 11.9 of the X.691 standard.

Table 2.150. Parameters

value	Value to be encoded

virtual void EncodeSmallNonNegWholeNumber (int value)

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

Table 2.151. Parameters

value	Value to be encoded

virtual long EncodeUnconsLength (long value)

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

Table 2.152. 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 fragementation was done (i.e the value was >= 16k).

override void HexDump ()

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

virtual bool IsAligned ()

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

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

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 SetAligned (bool value)

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

Table 2.153. Parameters

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

void SetSizeConstraint (long lower, long upper)

This method is used to set a size constraint within the buffer object. The constraint will only be set if an existing constraint is not already in place (i.e. if the existing reference is not null). This is used for length decoding of SEQUENCE OF objects to pass constraints applied to referenced objects to the base object.

void SetSizeConstraintExt (long lower, long upper, long extLower, long extUpper)

This method is used to set an extensible size constraint in the buffer object. The constraint will only be set if an existing constraint is not already in place (i.e. if the existing reference is not null). This is used for length encoding of SEQUENCE OF objects to pass constraints applied to referenced objects to the base object.

Asn1PerEncoder class Reference

Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHa class Reference

Private Attributes

- Asn1PerEncodeBuffer mMessageBuffer

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

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.

Definition at line 38 of file Asn1PerEncodeTraceHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1PerEncodeTraceHandler.cs

Asn1PerEncodeTraceHandler (Asn1PerEncodeBuffer messageBuffer)

This constructor initializes internal trace handler member variables.

Table 2.154. Parameters

messageBuffer	PER encode message buffer object reference

override void Enable ()

This method is used to turn PER bit tracing on

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

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

Table 2.155. Parameters

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

override void Reset ()

This method resets the trace bit field list.

Com::Objsys::Asn1::Runtime::Asn1PerInputStream class Reference

- 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)

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.

Definition at line 36 of file Asn1PerInputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1PerInputStream.cs

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.

Table 2.156. Parameters

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

virtual int Available ()

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

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

Table 2.157. Exceptions

System.SystemException	if an I/O error occurs.

virtual void Close ()

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

Table 2.158. 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 reseting the input stream position to current position.

virtual bool MarkSupported ()

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

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

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.

Table 2.159. Parameters

nbytes	Number of bytes to skip

Table 2.160. Exceptions

System.SystemException	an I/O error occurs.

Returns: . Skipped number of bytes

Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer interface Reference

-
-

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

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.

Definition at line 36 of file Asn1PerMessageBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1PerMessageBuffer.cs

void ByteAlign ()

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

System.IO.Stream GetInputStream ()

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

Returns: . Stream containing message

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`

Com::Objsys::Asn1::Runtime::Asn1PerOutputStream class Reference

- Asn1PerEncodeBuffer mBuffer

- Asn1PerOutputStreamTraceHandler mTraceHandler

Private Attributes

- int mBufferSize
- System.Collections.ArrayList mCaptureBufferList
-
-

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

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.

Definition at line 33 of file Asn1PerOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1PerOutputStream.cs

Asn1PerOutputStreamTraceHandler mTraceHandler

Variable holds the PER message trace handler

Definition at line 41 of file Asn1PerOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1PerOutputStream.cs

virtual void AddCaptureBuffer (System.IO.MemoryStream buffer)

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.

Table 2.161. Parameters

buffer	Buffer into which captured bytes are to be stored

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.

Table 2.162. 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.

Table 2.163. Parameters

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

virtual void BinDump (System.String varName)

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

Table 2.164. Parameters

varName	Name of top-level message object variable

virtual 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.

Table 2.165. Parameters

outs	StreamWriter object to which output should be written

varName	Name of top-level message object variable

virtual void ByteAlign ()

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)

This method encodes a single bit value.

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

Table 2.166. Parameters

value	Boolean value of bit to be encoded.

Table 2.167. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes a single bit value.

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

Table 2.168. Parameters

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

Table 2.169. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeBits (byte value, int nbits)

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.

Table 2.170. Parameters

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

Table 2.171. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes bit values from an array of octets.

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

Table 2.172. Parameters

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

Table 2.173. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes bit values from an array of octets.

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

Table 2.174. 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.

Table 2.175. 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)

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.

Table 2.176. 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)

Table 2.177. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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.

Table 2.178. Parameters

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

Table 2.179. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeConsWholeNumber (long adjusted-Value, long rangeValue)

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.

Table 2.180. Parameters

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

Table 2.181. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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.

Table 2.182. Parameters

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

Table 2.183. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeInt (long value, int nbits)

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.

Table 2.184. Parameters

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

Table 2.185. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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.

Table 2.186. 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.

Table 2.187. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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.

Table 2.188. 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.

Table 2.189. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual long EncodeLength (long value)

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.

Table 2.190. 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 fragementation was done (i.e the value was >= 16k).

Table 2.191. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes a constrained length determinant value.

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

Table 2.192. Parameters

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

Table 2.193. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeLengthEOM (long value)

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.

Table 2.194. Parameters

value	Original length value that was encoded.

Table 2.195. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Table 2.196. 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

Table 2.197. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeOIDLengthAndValue (int[] value)

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

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

Table 2.198. Parameters

value	Integer array containing arcs to encode

Table 2.199. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Table 2.200. 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

Table 2.201. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeRelOIDLengthAndValue (int[] value)

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

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

Table 2.202. Parameters

value	Integer array containing arcs to encode

Table 2.203. Exceptions

Asn1Exception	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeSmallLength (int value)

This method implements the rules to encode a normally small length as specified in section 11.9 of the X.691 standard.

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

Table 2.204. Parameters

value	Value to be encoded

Table 2.205. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

virtual void EncodeSmallNonNegWholeNumber (int value)

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.

Table 2.206. Parameters

value	Value to be encoded

Table 2.207. 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)

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.

Table 2.208. Parameters

buffer	Buffer in which captured bytes stored

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)`.

Table 2.209. Parameters

b	the binary data.

Table 2.210. Exceptions

System.SystemException	if an I/O error occurs.

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

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

Table 2.211. Parameters

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

Table 2.212. Exceptions

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

override void WriteByte (int b)

Writes the specified byte to this output stream.

Table 2.213. Parameters

b	the byte.

Table 2.214. Exceptions

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

override void WriteByte (byte b)

Writes the specified byte to this output stream.

Table 2.215. Parameters

b	the byte.

Table 2.216. Exceptions

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

void WriteBuffer (bool forceFlush)

Writes the mBuffer to the stream, if it is full or forced to flush.

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

Table 2.217. Parameters

forceFlush	Force flush the buffer.

Table 2.218. Exceptions

Asn1InvalidArgException	Any exception thrown by the underlying Asn1PerEncodeBuffer.

Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamT class Reference

Private Attributes

- System.IO.MemoryStream mCaptureBuffer
- Asn1PerOutputStream mMessageStream

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

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.

Definition at line 38 of file Asn1PerOutputStreamTraceHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1PerOutputStreamTraceHandler.cs

Asn1PerOutputStreamTraceHandler (Asn1PerOutputStream outs)

This constructor initializes the internal trace handler member variables.

Table 2.219. Parameters

outs	PER message stream object reference

override void Enable ()

This method is used to turn PER bit tracing on

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

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

Table 2.220. Parameters

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

override void Reset ()

This method does nothing here.

virtual void ResetTrace ()

This method resets the trace bit field list.

Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler class Reference

Private Attributes

- Asn1PerMessageBuffer mMessageBuffer

- Asn1PerBitFieldList mBitFieldList

-

- Asn1PerTraceHandler (Asn1PerMessageBuffer messageBuffer)

- virtual void AddElemName (System.String name, int arrayx)
- abstract void Enable ( )
- bool IsEnabled ( )
- virtual void NewBitField (System.String name, int bitCount)
- abstract void Print (System.IO.StreamWriter outs, System.String varName)
- virtual void RemoveLastElemName ( )
- void ReplaceLastFieldWithDetail (Asn1PerTraceHandler details)
- abstract void Reset ( )
- virtual void SetBitCount ( )
- virtual void SetBitOffset ( )

Detailed Description

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

Definition at line 35 of file Asn1PerTraceHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1PerTraceHandler.cs

Asn1PerBitFieldList mBitFieldList

Variable holds the bit field list. When not null, tracing is enabled.

Definition at line 40 of file Asn1PerTraceHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1PerTraceHandler.cs

Asn1PerTraceHandler (Asn1PerMessageBuffer messageBuffer)

This constructor initializes internal trace handler member variables.

Table 2.221. Parameters

messageBuffer	PER message buffer object reference

virtual void AddElemName (System.String name, int arrayx)

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).

Table 2.222. 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 ()

This method is used to turn PER bit tracing on or off. POST: mBitFieldList != null.

bool IsEnabled ()

Check whether bit tracing is on or off.

virtual void NewBitField (System.String name, int bitCount)

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

Table 2.223. 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)

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

Table 2.224. Parameters

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

virtual void RemoveLastElemName ()

This method removes the last element name int 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.

void ReplaceLastFieldWithDetail (Asn1PerTraceHandler details)

Replaces the last bit field in this bit trace handler with the fields from the given trace handler, which are assumed to be a breakdown of the last field.

Table 2.225. Parameters

details	handler to take details from

abstract void Reset ()

This method resets the trace bit field list.

virtual void SetBitCount ()

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 ()

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

Com::Objsys::Asn1::Runtime::Asn1PerUtil class Reference

- static int GetMsgBitCnt (int byteCount, int bitOffset)

Detailed Description

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

Definition at line 36 of file Asn1PerUtil.cs

The Documentation for this struct was generated from the following file:

- Asn1PerUtil.cs

static int GetMsgBitCnt (int byteCount, int bitOffset)

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

Table 2.226. 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

Com::Objsys::Asn1::Runtime::Asn1SetDuplicateExcep class Reference

- Asn1SetDuplicateException (Asn1BerDecodeBuffer buffer, Asn1Tag tag)
- Asn1SetDuplicateException (string message)

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..

Definition at line 36 of file Asn1SetDuplicateException.cs

The Documentation for this struct was generated from the following file:

- Asn1SetDuplicateException.cs

Asn1SetDuplicateException (Asn1BerDecodeBuffer buffer, Asn1Tag tag)

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

Table 2.227. Parameters

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

Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandl interface Reference

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

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.

Definition at line 41 of file Asn1TaggedEventHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1TaggedEventHandler.cs

void Contents (byte[] data)

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

Table 2.228. Parameters

data	Array containing encoded contents bytes.

void EndElement (Asn1Tag tag)

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

Table 2.229. Parameters

tag	Parsed tag value.

void StartElement (Asn1Tag tag, int len, byte[] tagLen-Bytes)

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

Table 2.230. Parameters

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

Com::Objsys::Asn1::Runtime::Asn1TagMatchFailedEx class Reference

- Asn1TagMatchFailedException (Asn1BerDecodeBuffer buffer, Asn1Tag expectedTag, Asn1Tag parsedTag)

- Asn1TagMatchFailedException (Asn1BerDecodeBuffer buffer, Asn1Tag expectedTag)

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..

Definition at line 36 of file Asn1TagMatchFailedException.cs

The Documentation for this struct was generated from the following file:

- Asn1TagMatchFailedException.cs

Asn1TagMatchFailedException (Asn1BerDecodeBuffer buffer, Asn1Tag expectedTag, Asn1Tag parsedTag)

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

Table 2.231. 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.

Table 2.232. Parameters

buffer	BER decode buffer object reference
expectedTag	Expected tag value

Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer class Reference

- XmlSource mInputSource

-

- Asn1XerDecodeBuffer (System.String source)

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 it to act as an input source for XML data to be read by a SAX parser.

Definition at line 36 of file Asn1XerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1XerDecodeBuffer.cs

XmlSource mInputSource

Variable holds the SAX input source object.

Definition at line 39 of file Asn1XerDecodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1XerDecodeBuffer.cs

Asn1XerDecodeBuffer (System.String source)

This constructor creates an XER decode buffer.

Table 2.233. Parameters

source	The source containing the XML document.

Com::Objsys::Asn1::Runtime::Asn1XerElemInfo class Reference

- int id
- System.String [] names
- bool optional
-
-
- Asn1XerElemInfo (System.String [] names, bool optional, int id)
- bool Matches (System.String name)

Detailed Description

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

Definition at line 32 of file Asn1XerElemInfo.cs

The Documentation for this struct was generated from the following file:

- Asn1XerElemInfo.cs

int id

id is the identifier used in generated code for the group

Definition at line 53 of file Asn1XerElemInfo.cs

The Documentation for this struct was generated from the following file:

- Asn1XerElemInfo.cs

System.String [] names

names[] is an array of all possible element names which may begin a component. A component may be a single XML element, in which case the array will be of size one. Or, a component may be a group of elements, a partial XML element content, with many possible first elements, in which case the array will have an entry for each of the first elements.

Definition at line 41 of file Asn1XerElemInfo.cs

The Documentation for this struct was generated from the following file:

- Asn1XerElemInfo.cs

bool optional

optional indicates whether the component is optional or not. In the case where the component is a partial XML element content (ie a group), this indicates whether the entire group is optional, not whether the individual first elements are optional.

Definition at line 49 of file Asn1XerElemInfo.cs

The Documentation for this struct was generated from the following file:

- Asn1XerElemInfo.cs

Asn1XerElemInfo (System.String[] names, bool optional, int id)

This constructor creates the element object.

Table 2.234. Parameters

names	first element names
optional	true if component having given names is optional
id	identifier

bool Matches (System.String name)

Determines whether this object matches the given element name.

Table 2.235. Parameters

name	The element name to check for a match.

Returns:. `true` if the given element name matches any of the names associated with this object; otherwise, `false`.

Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer class Reference

- bool mCanonical
- int mLevel
- System.String mLineSep
- int mState
-
-
- static Asn1RunTime rt
- Asn1XerEncodeBuffer ( )
- Asn1XerEncodeBuffer (bool canonical)
- Asn1XerEncodeBuffer (bool canonical, int sizeIncrement)
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- virtual void Copy (System.String 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)
- virtual void EncodeEndDocument ( )
- virtual void EncodeEndElement (System.String elemName)
- virtual void EncodeHexStrValue (byte [] data)
- virtual void EncodeNamedValue (System.String valueName, System.String elemName)
- virtual void EncodeNamedValueElement (System.String elemName)
- virtual void EncodeRealValue (double data, System.String elemName)
- virtual void EncodeStartDocument ( )
- virtual void EncodeStartElement (System.String elemName)
- virtual void IncrLevel ( )
- virtual void Indent ( )
- virtual bool IsCanonical ( )

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.

Definition at line 34 of file Asn1XerEncodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncodeBuffer.cs

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.

Table 2.236. 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.

Table 2.237. 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.

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.

Table 2.238. Parameters

outs	Output will be written to this stream
varName	Name of the Decoded ASN1 Type

virtual void Copy (System.String data)

This method copies a character string to the encode buffer.

Table 2.239. Parameters

data	The string value to copy

virtual void DecrLevel ()

This method decrements the element nesting level counter.

virtual void EncodeBinStrValue (byte[] bits, int nbits)

This method encodes XML binary string data

Table 2.240. Parameters

bits	Bit String to encode
nbits	Number of bits to encode

virtual void EncodeByte (byte data)

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.

Table 2.241. Parameters

data	The byte value to copy

virtual void EncodeData (System.String data)

This method encodes XML string data

Table 2.242. Parameters

data	String value to encode

virtual void EncodeEmptyElement (System.String elemName)

This method encodes an XML empty element tag

Table 2.243. Parameters

elemName	The name of element.

virtual void EncodeEndDocument ()

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

virtual void EncodeEndElement (System.String elemName)

This method encodes an XML end element tag

Table 2.244. Parameters

elemName	The name of element.

virtual void EncodeHexStrValue (byte[] data)

This method encodes XML hexadecimal string data

Table 2.245. Parameters

data	Data to encode

virtual void EncodeNamedValue (System.String value-Name, System.String elemName)

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

Table 2.246. Parameters

elemName	The name of element.
valueName	named value.

virtual void EncodeNamedValueElement (System.String elemName)

This method encodes an XML named value element tag

Table 2.247. Parameters

elemName	The name of element.

Table 2.248. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeRealValue (double data, System.String elemName)

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

Table 2.249. Parameters

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

Table 2.250. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeStartDocument ()

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

virtual void EncodeStartElement (System.String elemName)

This method encodes an XML start element tag

Table 2.251. Parameters

elemName	The name of element.

virtual void IncrLevel ()

This method increments the element nesting level counter.

virtual void Indent ()

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

Com::Objsys::Asn1::Runtime::Asn1XerEncoder interface Reference

•

- void EncodeEmptyElement (System.String elemName)
- void EncodeEndElement (System.String elemName)
- void EncodeNamedValue (System.String valueName, System.String elemName)
- void EncodeRealValue (double valueName, System.String elemName)
- void EncodeStartElement (System.String elemName)

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.

Definition at line 54 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

void EncodeEmptyElement (System.String elemName)

This method encodes an XML empty element tag.

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

Table 2.252. Parameters

elemName	The name of element.

Table 2.253. Exceptions

Asn1Exception	Thrown, if operation is failed.

void EncodeEndElement (System.String elemName)

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.

Table 2.254. Parameters

elemName	The name of element.

Table 2.255. Exceptions

Asn1Exception	Thrown, if operation is failed.

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.

Table 2.256. Parameters

valueName	The name of value.
elemName	The name of element.

Table 2.257. Exceptions

Asn1Exception	Thrown, if operation is failed.

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.

Table 2.258. Parameters

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

Table 2.259. Exceptions

Asn1Exception	Thrown, if operation is failed.

void EncodeStartElement (System.String elemName)

This method encodes an XML start element tag.

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

Table 2.260. Parameters

elemName	The name of element.

Table 2.261. Exceptions

Asn1Exception	Thrown, if operation is failed.

Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields struct Reference

- static readonly int XERDATA
- static readonly int XEREND
- static readonly int XERINDENT
- static readonly int XERINIT
- static readonly int XERSTART

Detailed Description

This class defines the constant variables for Asn1XerEncoder.

Definition at line 30 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

readonly int XERDATA

XER characters (data) state

Definition at line 42 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

readonly int XEREND

XER end element state

Definition at line 44 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

readonly int XERINDENT

Number of indent spaces required to print XER element

Definition at line 34 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

readonly int XERINIT

XER initialization state

Definition at line 38 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

readonly int XERSTART

XER start element state

Definition at line 40 of file Asn1XerEncoder.cs

The Documentation for this struct was generated from the following file:

- Asn1XerEncoder.cs

Com::Objsys::Asn1::Runtime::Asn1XerOpenType class Reference

- Asn1XerOpenType ( )
- Asn1XerOpenType (byte [] data)
- Asn1XerOpenType (byte [] data, int offset, int nbytes)
- Asn1XerOpenType (Asn1EncodeBuffer buffer)

  see Asn1OpenType(Asn1EncodeBuffer)

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. The data held in objects of this type should be UTF-8 encoded XML

Definition at line 41 of file Asn1XerOpenType.cs

The Documentation for this struct was generated from the following file:

- Asn1XerOpenType.cs

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. The data is assumed to be UTF-8 encoded XML. It should represent an XML encoding of the actual type.

Table 2.262. Parameters

data	Byte array containing a previously encoded value.

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. The data is assumed to be UTF-8 encoded XML. It should represent an XML encoding of the actual type.

Table 2.263. Parameters

data	Byte array containing a previously encoded value.
offset	The byte offset in array at which to begin.
nbytes	Number of bytes to copy from offset

Com::Objsys::Asn1::Runtime::Asn1XerOutputStream class Reference

- bool mCanonical
- int mLevel
- System.String mLineSep
- int mState
-
-
- static Asn1RunTime rt
- Asn1XerOutputStream (System.IO.Stream os)
- Asn1XerOutputStream (System.IO.Stream os, bool canonical, int bufSize)
- virtual void Copy (byte data)
- virtual void Copy (byte [] data)
- virtual void Copy (byte [] data, int off, int len)
- virtual void Copy (System.String 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)
- virtual void EncodeEndDocument ( )
- virtual void EncodeEndElement (System.String elemName)

- virtual void EncodeHexStrValue (byte [] data)
- virtual void EncodeNamedValue (System.String valueName, System.String elemName)
- virtual void EncodeNamedValueElement (System.String elemName)
- virtual void EncodeRealValue (double data, System.String elemName)
- virtual void EncodeStartDocument ( )
- virtual void EncodeStartElement (System.String elemName)
- virtual void IncrLevel ( )
- virtual void Indent ( )
- virtual bool IsCanonical ( )
- virtual void Write (System.String data)

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.

Definition at line 38 of file Asn1XerOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1XerOutputStream.cs

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.

Table 2.264. 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.

Table 2.265. 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.

virtual void Copy (byte data)

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.

Table 2.266. Parameters

data	The byte value to copy

virtual void Copy (byte[] data)

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.

Table 2.267. Parameters

data	Array of bytes to copy to the output stream

Table 2.268. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void Copy (byte[] data, int off, int len)

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.

Table 2.269. 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

Table 2.270. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void Copy (System.String data)

This method copies a character string to the output stream.

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

Table 2.271. Parameters

data	The string value to copy

Table 2.272. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void DecrLevel ()

This method decrements the element nesting level counter.

virtual void EncodeBinStrValue (byte[] bits, int nbits)

This method encodes XML binary string data

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

Table 2.273. Parameters

bits	Bit String to encode
nbits	Number of bits to encode

Table 2.274. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeByte (byte data)

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.

Table 2.275. Parameters

data	The byte value to copy

virtual void EncodeData (System.String data)

This method encodes XML string data

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

Table 2.276. Parameters

data	String value to encode

Table 2.277. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeEmptyElement (System.String elemName)

This method encodes an XML empty element tag.

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

Table 2.278. Parameters

elemName	The name of element.

Table 2.279. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeEndDocument ()

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

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

Table 2.280. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeEndElement (System.String elemName)

This method encodes an XML end element tag.

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

Table 2.281. Parameters

elemName	The name of element.

Table 2.282. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeHexStrValue (byte[] data)

This method encodes XML hexadecimal string data

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

Table 2.283. Parameters

data	Data to encode

Table 2.284. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeNamedValue (System.String valueName, System.String elemName)

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

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

Table 2.285. Parameters

valueName	The named value.

elemName	The name of element.

Table 2.286. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeNamedValueElement (System.String elemName)

This method encodes an XML named value element tag.

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

Table 2.287. Parameters

elemName	The name of element.

Table 2.288. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeRealValue (double data, System.String elemName)

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

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

Table 2.289. Parameters

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

Table 2.290. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeStartDocument ()

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

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

Table 2.291. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeStartElement (System.String elemName)

This method encodes an XML start element tag.

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

Table 2.292. Parameters

elemName	The name of element.

Table 2.293. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void IncrLevel ()

This method increments the element nesting level counter.

virtual void Indent ()

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.

Table 2.294. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void Write (System.String data)

This method copies a character string to the output stream.

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

Table 2.295. Parameters

data	The string value to copy

Table 2.296. Exceptions

Asn1Exception	Thrown, if operation is failed.

Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler class Reference

- int mCurrElemID
- int mCurrState
- int mLevel
- string mXMLElemName
- readonly int XERDATA
- readonly int XEREND
- readonly int XERINIT
- readonly int XERSTART
- readonly int XERUNKNOWN

Private Attributes

- bool mGroupComplete
- int mStartLevel

Protected Attributes

- bool mConsumedStartElement
-
-
- bool ConsumeStartElement (String namespaceURI, String localName, String qName, XmlAttributes atts)

  <summary> This method should be used in preference to invoking StartElement directly when a parent SAX handler is delegating to a child SAX handler.
- virtual void EndGroup ( )

<summary>

- override void Error (System.Xml.XmlException exception)
- override void FatalError (System.Xml.XmlException exception)
- virtual void Init (int startLevel)
- bool IsComplete ( )
- bool IsDecodingAsGroup ( )

  <summary> Return true if this SAX handler is decoding a group of elements rather than a single element (and possibly its children elements).

- bool MatchXMLElemName (string elemName)
- void SetComplete ( )
- void SetXMLElemName (string value)
- override void Warning (System.Xml.XmlException exception)

- Asn1XerSaxHandler ( )

Detailed Description

This class extends the DefaultHandler SAX handler class to add items specific to ASN.1 XER encoding.

Definition at line 31 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

int mCurrElemID

Variable holds the current element ID

Definition at line 46 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

int mCurrState

Variable holds the current XER processing state

Definition at line 44 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

int mLevel

Variable holds the start and current level

Definition at line 66 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

readonly int XERDATA

XER characters (data) state

Definition at line 39 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

readonly int XEREND

XER end element state

Definition at line 41 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

readonly int XERINIT

XER initialization state

Definition at line 35 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

readonly int XERSTART

XER start element state

Definition at line 37 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

readonly int XERUNKNOWN

XER unknown state

Definition at line 33 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

Member Data Documentation

bool mGroupComplete

Flag used to track when setComplete has been called and the endGroup event has been fired.

Definition at line 77 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

Member Data Documentation

bool mConsumedStartElement

Flag used to signal whether startElement actually handled the given element. See the description of method consumeStartElement

Definition at line 72 of file Asn1XerSaxHandler.cs

The Documentation for this struct was generated from the following file:

- Asn1XerSaxHandler.cs

bool ConsumeStartElement (String namespaceURI, String localName, String qName, XmlAttributes atts)

Using this method gives the StartElement method the opportunity to set the mConsumedStartEvent flag to false to signal that the given element does not belong to the group and that the group is complete. This method invokes the StartElement event and returns the resulting value of the mConsumedStartElement flag (true, unless the StartElement method explicitly sets it to false). If mConsumedStartElement is set to false, this method will invoke SetComplete, marking the handler complete and triggering the EndGroup event, if it has not already fired.

If the StartElement method is certain that some other element is required instead of the one given, it is preferable to throw Asn1XmlSaxUnexpElemExc to indicate this. Otherwise, if the given element does not belong to the group being decoded, mConsumedStartElement can be set false and the element ignored. It is then up to the parent SAX handler to recognize the element as part of a different group or report it as an unexpected element. The StartElement method should not set mConsumedStartEvent false except when mLevel <= mStartLevel, since this is a precondition for SetComplete.

Returns: . true if the StartElement method consumed the given element or false if not.

Table 2.297. Exceptions

Asn1XmlSaxUnexpElemExc	if certain that some other element is expected.

virtual void EndGroup ()

This event method should be invoked when the group being decoded by this handler is decided to be complete. Subclasses may implement this event method to do things such as check for missing required elements. This event will be triggerd, when appropriate, by ConsumeStartElement. Subclasses may invoke it directly when appropriate, but the preferred method is to invoke it indirectly by invoking SetComplete.

override void Error (System.Xml.XmlException exception)

This method manage when an error exception ocurrs in the parsing process

Table 2.298. Parameters

exception	The exception throws by the parser

Table 2.299. Exceptions

System.Xml.XmlException	The error exception

override void FatalError (System.Xml.XmlException exception)

This method throws a fatal error exception.

Table 2.300. Parameters

exception	The exception thrown by the parser

Table 2.301. Exceptions

System.Xml.XmlException	The error exception

virtual void Init (int startLevel)

This method initializes the class member variables.

Table 2.302. Parameters

startLevel	The XER level to begin

bool IsDecodingAsGroup ()

void SetComplete ()

Invoke this method to mark this SAX handler as being complete. This method will trigger the EndGroup event, if it has not already fired. This is the preferred way for a subclass to trigger the EndGroup event, rather than to invoke EndGroup directly (mainly for consistency). Subclasses should consider invoking this method in the EndElement event when either of the following conditions obtain:

- mLevel returns to 0 (indicating the group must be complete)
- The group is self-delimiting and known to be complete on that basis. This method should only be invoked when mLevel <= mStartLevel. Otherwise, the handler could not possibly be complete (note that mLevel == mStartLevel does not entail the handler is complete, because this is a normal state of affairs between elements when decoding a group).

Table 2.303. Exceptions

InvalidOperationException	if mLevel != mStartLevel.

override void Warning (System.Xml.XmlException exception)

This method manage when a warning exception ocurrs in the parsing process

Table 2.304. Parameters

exception	The exception Throws by the parser

Table 2.305. Exceptions

System.Xml.XmlException	The warning exception

Asn1XerSaxHandler ()

The default constructor creates an XER SAX parser instance.

Com::Objsys::Asn1::Runtime::Asn1XerUtil class Reference

- static void EncodeReal (Asn1XerEncoder buffer, double value, System.String elemName)

Detailed Description

This class contains some general purpose static utility functions for XER encoding or decoding.

Definition at line 33 of file Asn1XerUtil.cs

The Documentation for this struct was generated from the following file:

- Asn1XerUtil.cs

static void EncodeReal (Asn1XerEncoder buffer, double value, System.String elemName)

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

Table 2.306. Parameters

buffer	Encode message buffer object
value	Value to be encoded.
elemName	Element name

Com::Objsys::Asn1::Runtime::Asn1XmlEncodeBuffer class Reference

- bool mCanonical
- Asn1XmlEncodeHelper mHelper
- int mIndent
- int mLevel
- System.String mLineSep
- int mState
- Asn1XmlXSIAttrs mXSIAttrs
- IList openElemNsDecl
-
-
-
- static Asn1RunTime rt

- Asn1XmlEncodeBuffer ( )
- Asn1XmlEncodeBuffer (int sizeIncrement)
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- virtual void Copy (System.String data)
- virtual void DecrLevel ( )
- virtual void EncodeAttr (System.String qname, System.String data)
- virtual void EncodeBase64BinaryAttribute (byte [] data, System.String attrName)
- virtual void EncodeBinStrValue (byte [] bits, int nbits)
- virtual void EncodeBoolAttribute (bool data, System.String attrName)
- virtual void EncodeByte (byte data)
- virtual void EncodeData (System.String data)
- virtual void EncodeDoubleValue (double data, System.String elemName, System.String nsPrefix)
- virtual void EncodeEmptyElement (System.String elemName, System.String nsPrefix, bool terminate)
- virtual void EncodeEndDocument ( )
- virtual void EncodeEndElement (System.String elemName, System.String nsPrefix)
- virtual void EncodeEndElement (System.String elemName, System.String nsPrefix, bool indent)
- virtual void EncodeHexStrValue (byte [] data)
- virtual void EncodeIntAttribute (int data, System.String attrName)
- virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String nsPrefix)
- virtual void EncodeNamedValueElement (System.String valueName)
- void EncodeNamespace (Asn1XmlNamespace ns)

  Encode the given namespace on the next start element tag.
- virtual void EncodeStartDocument ( )
- virtual void EncodeStartElement (System.String elemName, System.String nsPrefix, bool terminate)
- virtual void EncodeXSIAttrs ( )
- virtual void IncrLevel ( )
- virtual void Indent ( )
- virtual bool IsCanonical ( )
- virtual void SetXSIAttrs (Asn1XmlXSIAttrs data)

- void WriteNamespaces ( )

  This writes the ns declarations for the namespaces in openElemNsDecl.

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).

Definition at line 35 of file Asn1XmlEncodeBuffer.cs

The Documentation for this struct was generated from the following file:

- Asn1XmlEncodeBuffer.cs

Asn1XmlEncodeBuffer ()

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

Asn1XmlEncodeBuffer (int sizeIncrement)

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

Table 2.307. 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.

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.

Table 2.308. Parameters

outs	Output will be written to this stream
varName	Name of the Decoded ASN1 Type

virtual void Copy (System.String data)

This method copies a character string to the encode buffer.

Table 2.309. Parameters

data	The string value to copy

virtual void DecrLevel ()

This method decrements the element nesting level counter.

virtual void EncodeAttr (System.String qname, System.String data)

This method encodes an XML attribute value.

Table 2.310. Parameters

qname	Attribute qualified name.
value	Attribute value in string form.

virtual void EncodeBinStrValue (byte[] bits, int nbits)

This method encodes XML binary string data

Table 2.311. Parameters

bits	Bit string to encode

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeByte (byte data)

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.

Table 2.312. Parameters

data	The byte value to copy

virtual void EncodeData (System.String data)

This method encodes XML string data

Table 2.313. Parameters

value	String value to encode

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeDoubleValue (double data, System.String elemName, System.String nsPrefix)

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

Table 2.314. Parameters

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

Table 2.315. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeEmptyElement (System.String elemName, System.String nsPrefix, bool terminate)

This method encodes an XML empty element tag

Table 2.316. Parameters

elemName	The name of element.
nsPrefix	The namespace prefix of element.

<throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeEndDocument ()

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

virtual void EncodeEndElement (System.String elemName, System.String nsPrefix)

This method encodes an XML end element tag

Table 2.317. Parameters

elemName	The name of element, as String.

virtual void EncodeEndElement (System.String elemName, System.String nsPrefix, bool indent)

This method encodes an XML end element tag without indenting

Table 2.318. Parameters

elemName	The name of element.

<throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeHexStrValue (byte[] data)

This method encodes XML hexadecimal string data

Table 2.319. Parameters

data	Data to encode

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String nsPrefix)

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

virtual void EncodeNamedValueElement (System.String valueName)

This method encodes an XML named value element tag

Table 2.320. Parameters

valueName	The named value, as String.

Table 2.321. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeStartDocument ()

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

virtual void EncodeStartElement (System.String elemName, System.String nsPrefix, bool terminate)

This method encodes an XML start element tag.

Table 2.322. Parameters

elemName	The name of element.
nsPrefix	The namespace prefix of element.

<throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeXSIAttrs ()

This method encodes XSI attributes.

virtual void IncrLevel ()

This method increments the element nesting level counter.

virtual void Indent ()

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

virtual void SetXSIAttrs (Asn1XmlXSIAttrs data)

This method sets the XSI attributes object to the given value.

Table 2.323. Parameters

value	XSI attributes object

Asn1XmlEncoder class Reference

Com::Objsys::Asn1::Runtime::Asn1XmlOutputStream class Reference

- bool mCanonical
- Asn1XmlEncodeHelper mHelper

- int mIndent
- int mLevel
- System.String mLineSep
- int mState
- Asn1XmlXSIAttrs mXSIAttrs
- IList openElemNsDecl
-
-
-
- static Asn1RunTime rt
- Asn1XmlOutputStream (System.IO.Stream os)
- Asn1XmlOutputStream (System.IO.Stream os, int bufSize)
- Asn1XmlOutputStream (System.IO.Stream os, bool canonical, int bufSize)
- virtual void Copy (byte data)
- virtual void Copy (byte [] data)
- virtual void Copy (byte [] data, int off, int len)
- virtual void Copy (System.String data)
- virtual void DecrLevel ( )
- virtual void EncodeAttr (System.String qname, System.String data)
- virtual void EncodeBinStrValue (byte [] bits, int nbits)
- virtual void EncodeByte (byte data)
- virtual void EncodeData (System.String data)
- virtual void EncodeDoubleValue (double data, System.String elemName, System.String nsPrefix)
- virtual void EncodeEmptyElement (System.String elemName, System.String nsPrefix, bool terminate)
- virtual void EncodeEndDocument ( )
- virtual void EncodeEndElement (System.String elemName, System.String nsPrefix)
- virtual void EncodeEndElement (System.String elemName, System.String nsPrefix, bool indent)
- virtual void EncodeHexStrValue (byte [] data)
- virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String nsPrefix)

- virtual void EncodeNamedValueElement (System.String valueName)
- void EncodeNamespace (Asn1XmlNamespace ns)

  Encode the given namespace on the next start element tag.
- virtual void EncodeStartDocument ( )
- virtual void EncodeStartElement (System.String elemName, System.String nsPrefix, bool terminate)
- virtual void EncodeXSIAttrs ( )
- virtual void IncrLevel ( )
- virtual void Indent ( )
- virtual bool IsCanonical ( )
- virtual void SetXSIAttrs (Asn1XmlXSIAttrs data)
- virtual void Write (System.String data)
- void WriteNamespaces ( )

  This writes the ns declarations for the namespaces in openElemNsDecl.

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.

Definition at line 38 of file Asn1XmlOutputStream.cs

The Documentation for this struct was generated from the following file:

- Asn1XmlOutputStream.cs

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.

Table 2.324. 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.

Table 2.325. 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.

Table 2.326. 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.

virtual void Copy (byte data)

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.

Table 2.327. Parameters

value	The byte value to copy

<throws> IOException Any exception thrown by the underlying OutputStream. </throws>

virtual void Copy (byte[] data)

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).

Table 2.328. Parameters

value	Array of bytes to copy to the output stream

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void Copy (byte[] data, int off, int len)

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).

Table 2.329. Parameters

value	Array of bytes to copy to the output stream
off	Starting offset in array
len	The length to be encoded

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void Copy (System.String data)

This method copies a character string to the output stream.

Table 2.330. Parameters

value	The string value to copy

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void DecrLevel ()

This method decrements the element nesting level counter.

virtual void EncodeAttr (System.String qname, System.String data)

This method encodes an XML attribute value.

Table 2.331. Parameters

qname	Attribute qualified name.
value	Attribute value in string form.

virtual void EncodeBinStrValue (byte[] bits, int nbits)

This method encodes XML binary string data

Table 2.332. Parameters

bits	Bit string to encode

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeByte (byte data)

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.

Table 2.333. Parameters

value	The byte value to copy

<throws> IOException Any exception thrown by the underlying OutputStream. </throws>

virtual void EncodeData (System.String data)

This method encodes XML string data

Table 2.334. Parameters

value	String value to encode

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeDoubleValue (double data, System.String elemName, System.String nsPrefix)

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

Table 2.335. Parameters

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

Table 2.336. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeEmptyElement (System.String elemName, System.String nsPrefix, bool terminate)

This method encodes an XML empty element tag

Table 2.337. Parameters

elemName	The name of element.
nsPrefix	The namespace prefix of element.

<throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeEndDocument ()

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

virtual void EncodeEndElement (System.String elemName, System.String nsPrefix)

This method encodes an XML end element tag

Table 2.338. Parameters

elemName	The name of element, as String.

virtual void EncodeEndElement (System.String elemName, System.String nsPrefix, bool indent)

This method encodes an XML end element tag without indenting

Table 2.339. Parameters

elemName	The name of element.

<throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeHexStrValue (byte[] data)

This method encodes XML hexadecimal string data

Table 2.340. Parameters

data	Data to encode

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String nsPrefix)

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

Table 2.341. Parameters

valueName	The name of value.
elemName	The name of element.

<throws> IOException If I/O error occurs. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeNamedValueElement (System.String valueName)

This method encodes an XML named value element tag

Table 2.342. Parameters

valueName	The named value, as String.

Table 2.343. Exceptions

Asn1Exception	Thrown, if operation is failed.

virtual void EncodeStartDocument ()

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

virtual void EncodeStartElement (System.String elemName, System.String nsPrefix, bool terminate)

This method encodes an XML start element tag.

Table 2.344. Parameters

elemName	The name of element.
nsPrefix	The namespace prefix of element.

<throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void EncodeXSIAttrs ()

This method encodes XSI attributes.

virtual void IncrLevel ()

This method increments the element nesting level counter.

virtual void Indent ()

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

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

virtual void SetXSIAttrs (Asn1XmlXSIAttrs data)

This method sets the XSI attributes object to the given value.

Table 2.345. Parameters

value	XSI attributes object

virtual void Write (System.String data)

This method copies a character string to the output stream.

Table 2.346. Parameters

value	The string value to copy

<throws> IOException Any exception thrown by the underlying OutputStream. </throws> <throws> Asn1Exception Thrown, if operation is failed. </throws>

Com::Objsys::Asn1::Runtime::Asn1XmlUtil class Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XmlUtil::CaptureElementTracking

- static bool mKeepNulls

- static string CaptureElement (System.Xml.XmlReader reader, bool contentOnly, bool injectNsDecls)
- static Asn1Type DecodeXerOpenType (Asn1OpenTypeField openType, System.Xml.XmlReader reader)
- static void EncodeDouble (Asn1XmlEncoder buffer, double data, System.String elemName, System.String nsPrefix)
- static void EncodeDouble (Asn1XmlEncoder buffer, double data)
- static void EncodeNSAttrs (Asn1XmlEncoder buffer, Asn1XmlNamespace [] nsArray)
- static double GetMinusZero ( )
- static string GetTextContent (System.Xml.XmlReader reader)

  Return the current text node and subsequent text content until an EndElement is found.
- static System.String GetXMLString (System.String data)

  <summary> This method will convert the given string value into XML character content by escaping special characters in the sting such as ampersand (&), left angle bracket (>), etc.
- static bool IsMinusZero (double value)
- static void KeepNullsInString (bool keep)
- static String [] TokenizeXsdList (String listValue)

- static void AppendEscaped (String value, System.Text.StringBuilder buf, bool forAttr)
- static void AppendQualifiedName (String prefix, String localName, System.Text.StringBuilder buf)
- static bool CollectionContains (ICollection< Asn1XmlNamespace > collection, String prefix)
- static void EndOfElement (Stack< CaptureElementTracking > trackingStack, LinkedList< Asn1XmlNamespace > nsDecls, int level, bool contentOnly, System.Text.StringBuilder buf, int nsDeclInjectionLoc)
- static String [] TokenizeXsdList (String listValue, String [] targetArray)

Detailed Description

This class contains some general purpose static utility functions for XML encoding or decoding.

Definition at line 35 of file Asn1XmlUtil.cs

The Documentation for this struct was generated from the following file:

- Asn1XmlUtil.cs

static string CaptureElement (System.Xml.XmlReader reader, bool contentOnly, bool injectNsDecls)

Capture the XML text for an element, returning it in a string object. Optionally, this method can capture the element's content only.

PRE: The reader is positioned on the start tag of the element to be captured, call it E. POST: The reader is positioned on the event following element E.

Table 2.347. Parameters

contentOnly	if TRUE, only capture the content of the current element. Otherwise, captures the current element's tags, namespace declarations, and attributes, as well as the content.
injectNsDecls	if TRUE, the outermost element(s) should include namespace declarations for prefixes used by those elements but not declared.

static void EncodeDouble (Asn1XmlEncoder buffer, double data, System.String elemName, System.String nsPrefix)

This method encodes an ASN.1 real value using the XML encoding (non-XER).

Table 2.348. Parameters

buffer	Encode message buffer object
value	Value to be encoded.
elemName	Element name
nsPrefix	Element namespace prefix value

static void EncodeDouble (Asn1XmlEncoder buffer, double data)

This method encodes an ASN.1 real value using the XML encoding (non-XER).

Table 2.349. Parameters

buffer	Encode message buffer object
value	Value to be encoded.

static void EncodeNSAttrs (Asn1XmlEncoder buffer, Asn1XmlNamespace[] nsArray)

This method encodes XML namespace attributes in the form 'xmlns[:prefix]="uri"'.

Table 2.350. Parameters

nsArray	Array of XML namespace prefix/URI mappings.

static double GetMinusZero ()

This method returns double value for "minus zero" (-0) special XML value.

Returns: . double value for "-0".

static string GetTextContent (System.Xml.XmlReader reader)

This will throw an exception if any Element nodes are encountered. PRE: The reader is positioned on a Text or CDATA event. POST: The reader is positioned just past the closing EndElement.

static System.String GetXMLString (System.String data)

Table 2.351. Parameters

data	String to convert

Returns: . Converted string value

static bool IsMinusZero (double value)

This method will return true, if value is "minus zero" (-0) special XML value.

Table 2.352. Parameters

value	value to test

Returns: . true, if this value is "-0".

static void KeepNullsInString (bool keep)

This method allows users to toggle the output of a null entity code in XML strings. Null bytes are not permitted in any XML document, but are permissible in ASN.1 strings. When converting, users may elect to retain null bytes as entities (&#x0;) by passing `true` to this function. By default, null bytes are dropped.

Table 2.353. Parameters

keep	Boolean switch to keep or ignore null bytes.

static String [] TokenizeXsdList (String listValue)

Return an array of strings representing the (lexical) values of an xsd:list

Table 2.354. Parameters

listValue	the lexical value of the xsd:list. It need not have whitepspace collapse applied yet.

Returns: . array of lexical values, one per value in the listValue. No empty strings will appear in the array.

static void AppendEscaped (String value, System.Text.StringBuilder buf, bool forAttr)

Append given string to the given buffer, with character escaping applied as required in XML attribute values and/or XML element text. For attributes, double quotes are escaped; single quotes are not.

Table 2.355. Parameters

value	The unescaped text.
buf	The buffer to append to.
forAttr	TRUE if for attribute value; FALSE otherwise.

static void AppendQualifiedName (String prefix, String localName, System.Text.StringBuilder buf)

Append given qualified name to buffer.

Table 2.356. Parameters

prefix	The prefix. If empty or null, no prefix is appended.
localName	The name to append.
buf	The buffer to append to.

static bool CollectionContains (ICollection< Asn1XmlNamespace > collection, String prefix)

Return true if the given collection has a namespace that has the given prefix.

static void EndOfElement (Stack< CaptureElementTracking > trackingStack, LinkedList< Asn1XmlNamespace > nsDecls, int level, bool contentOnly, System.Text.StringBuilder buf, int nsDeclInjectionLoc)

End of element processing for namespace injection for CaptureElement.

Table 2.357. Parameters

trackingStack	Stack of tracking information, with the top of the stack being the entry for the element whose end we have reached.
nsDecls	List of namespace declarations, which begins with prefixes used by the element and its children.
level	Level == 0 if the outermost element is being ended.
conentOnly	True if the capturing is only capturing the outermost element's content, and not the outermost element itself.
buf	Buffer into which content is being captured.
nsDeclInjectionLoc	Location in buffer where namespace declarations should be injected.

static String [] TokenizeXsdList (String listValue, String[] targetArray)

Tokenize an xsd:list value into the given array. If the given array is a null reference, then a new array of the correct size will be created (after determining the correct size) and this method will be recursively invoked to populate that array.

Table 2.358. Parameters

listValue	**Table 2.359. Parameters** targetArray The array into which the values are populated.

Asn1XmlXerEncoder class Reference

Com::Objsys::Asn1::Runtime::Asn1XmlUtil::CaptureEl class Reference

Private Attributes

- int additions
- List< Asn1XmlNamespace > declaredNamespaces
-
- void AddDeclaredNamespace (Asn1XmlNamespace ns)
- CaptureElementTracking (int additions)

- bool IsDeclared (String prefix)

Detailed Description

Class used by CaptureElement for tracking namespace declarations that have been made and that are used.

Definition at line 309 of file Asn1XmlUtil.cs

The Documentation for this struct was generated from the following file:

- Asn1XmlUtil.cs

bool IsDeclared (String prefix)

Return true if a namespace with the given prefix was added using AddDeclaredNamespace

Com::Objsys::Asn1::Runtime::XmlAttributes::XmlAttrib class Reference

Public Attributes

- System.String att_fullName
- System.String att_localName
- System.String att_type
- System.String att_URI
- System.String att_value

- XmlAttribute (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)

Detailed Description

This class is created to save the information of each attributes in the XmlAttributes.

Definition at line 391 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

Member Data Documentation

System.String att_fullName

Variable holds attribte full name (namespace + local name)

Definition at line 397 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

System.String att_localName

Variable holds attribte local name

Definition at line 395 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

System.String att_type

Variable holds attribte type

Definition at line 399 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

System.String att_URI

Variable holds attribte namespace

Definition at line 393 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

System.String att_value

Variable holds attribte value

Definition at line 401 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

XmlAttribute (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)

This is the constructor of the XmlAttribute

Table 2.360. 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

Com::Objsys::Asn1::Runtime::XmlAttributes class Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlAttributes::XmlAttribute

Private Attributes

- System.Collections.ArrayList MainList

- 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 Qname)
- virtual int GetIndex (System.String Uri, System.String Lname)
- virtual int GetLength ( )
- virtual System.String GetLocalName (int index)
- virtual System.String GetQName (int index)
- virtual System.String GetType (int index)
- virtual System.String GetType (System.String Qname)
- virtual System.String GetType (System.String Uri, System.String Lname)
- virtual System.String GetURI (int index)
- virtual System.String GetValue (int index)
- virtual System.String GetValue (System.String Qname)
- virtual System.String GetValue (System.String Uri, System.String Lname)
- virtual void RemoveAttribute (int index)
- virtual void RemoveAttribute (System.String indexName)
- 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 (XmlAttributes arrayList)

Detailed Description

This class will manage all the parsing operations emulating the SAX parser behavior

Definition at line 34 of file XmlAttributes.cs

The Documentation for this struct was generated from the following file:

- XmlAttributes.cs

virtual void Add (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)

Adds a new attribute elment to the given XmlAttributes instance.

Table 2.361. 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 ()

Clears the list of attributes in the given AttributesSupport instance.

virtual System.String GetFullName (int index)

Returns the qualified name of the attribute indicated by the given index.

Table 2.362. Parameters

index	The attribute index.

Returns: . The qualified name or empty string if the index is out of bounds.

virtual int GetIndex (System.String Qname)

Obtains the index of an attribute of the AttributeSupport from its qualified (long) name.

Table 2.363. 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 GetIndex (System.String Uri, System.String Lname)

Obtains the index of an attribute of the AttributeSupport from its namespace URI and its localname.

Table 2.364. 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 GetLength ()

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)

Returns the local name of the attribute in the given XmlAttributes instance that indicates the given index.

Table 2.365. 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 GetQName (int index)

Returns the qualified name of the attribute indicated by the given index. This is an alias for GetFullName.

Table 2.366. Parameters

index	

Returns: . The qualified name or empty string if the index is out of bounds.

virtual System.String GetType (int index)

Returns the type of the attribute in the given XmlAttributes instance that indicates the given index.

Table 2.367. 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 GetType (System.String Qname)

Returns the type of the Attribute that match with the given qualified name.

Table 2.368. 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 (System.String Uri, System.String Lname)

Returns the type of the Attribute that match with the given namespace URI and local name.

Table 2.369. 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 GetURI (int index)

Returns the namespace URI of the attribute in the given XmlAttributes instance that indicates the given index.

Table 2.370. 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 (int index)

Returns the value of the attribute in the given XmlAttributes instance that indicates the given index.

Table 2.371. 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 System.String GetValue (System.String Qname)

Returns the value of the Attribute that match with the given qualified name.

Table 2.372. 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 (System.String Uri, System.String Lname)

Returns the value of the Attribute that match with the given namespace URI and local name.

Table 2.373. 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 void RemoveAttribute (int index)

This method eliminates the XmlAttribute instance at the specified index.

Table 2.374. Parameters

index	The index of the attribute.

virtual void RemoveAttribute (System.String indexName)

This method eliminates the XmlAttribute instance in the specified index.

Table 2.375. Parameters

indexName	The index name of the attribute.

virtual void SetAttribute (int index, System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)

Replaces an XmlAttribute in the given XmlAttributes instance.

Table 2.376. 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)

Replaces all the list of XmlAttribute of the given XmlAttributes instance.

Table 2.377. Parameters

Source	The source XmlAttributes instance.

virtual void SetFullName (int index, System.String Full-Name)

Modifies the qualified name of the attribute in the given XmlAttributes instance.

Table 2.378. Parameters

index	The attribute index.

FullName	The new qualified name for the attribute.

virtual void SetLocalName (int index, System.String LocalName)

Modifies the local name of the attribute in the given XmlAttributes instance.

Table 2.379. Parameters

index	The attribute index.
LocalName	The new Local name for the attribute.

virtual void SetType (int index, System.String Type)

Modifies the type of the attribute in the given XmlAttributes instance.

Table 2.380. Parameters

index	The attribute index.
Type	The new type for the attribute.

virtual void SetURI (int index, System.String URI)

Modifies the namespace URI of the attribute in the given XmlAttributes instance.

Table 2.381. Parameters

index	The attribute index.
URI	The new namespace URI for the attribute.

virtual void SetValue (int index, System.String Value)

Modifies the value of the attribute in the given XmlAttributes instance.

Table 2.382. Parameters

index	The attribute index.
Value	The new value for the attribute.

XmlAttributes ()

Builds a new instance of XmlAttributes.

XmlAttributes (XmlAttributes arrayList)

Creates a new instance of XmlAttributes from an ArrayList of XmlAttribute class.

Table 2.383. Parameters

arrayList	An ArraList of XmlAttribute class instances.

Returns: . A new instance of XmlAttributes

Com::Objsys::Asn1::Runtime::XmlSaxContentHandler interface Reference

- 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)

Detailed Description

This interface will manage the Content events of a XML document.

Definition at line 33 of file XmlSaxContentHandler.cs

The Documentation for this struct was generated from the following file:

- XmlSaxContentHandler.cs

void Characters (char[] ch, int start, int length)

This method manage the notification when Characters elements were found.

Table 2.384. 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 EndDocument ()

This method manage the notification when the end document node were found.

void EndElement (System.String namespaceURI, System.String localName, System.String qName)

This method manage the notification when the end element node was found.

Table 2.385. Parameters

namespaceURI	The namespace URI of the element.
localName	The local name of the element.
qName	The long (qualified) name of the element.

void EndPrefixMapping (System.String prefix)

This method manage the event when an area of expecific URI prefix was ended.

Table 2.386. Parameters

prefix	The prefix that ends.

void IgnorableWhitespace (char[] Ch, int Start, int Length)

This method manage the event when a ignorable whitespace node was found.

Table 2.387. Parameters

Ch	The array with the ignorable whitespaces.
Start	The index in the array with the ignorable whitespace.
Length	The length of the whitespaces.

void ProcessingInstruction (System.String target, System.String data)

This method manage the event when a processing instruction was found.

Table 2.388. Parameters

target	The processing instruction target.
data	The processing instruction data.

void SetDocumentLocator (XmlSaxLocator locator)

This method is not supported, it is included for compatibility.

Table 2.389. Parameters

locator	A `XmlSaxLocator` object that can return the location of any events into the XML document

void SkippedEntity (System.String name)

This method manage the event when a skipped entity was found.

Table 2.390. Parameters

name	The name of the skipped entity.

void StartDocument ()

This method manage the event when a start document node was found.

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.

Table 2.391. 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.

void StartPrefixMapping (System.String prefix, System.String uri)

This methods indicates the start of a prefix area in the XML document.

Table 2.392. Parameters

prefix	The prefix of the area.
uri	The namespace URI of the prefix area.

Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler class Reference

- 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)

Detailed Description

This class provides the base implementation for the management of XML documents parsing.

Definition at line 37 of file XmlSaxDefaultHandler.cs

The Documentation for this struct was generated from the following file:

- XmlSaxDefaultHandler.cs

virtual void Characters (char[] ch, int start, int length)

This method manage the notification when Characters element were found.

Table 2.393. 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

virtual void EndDocument ()

This method manage the notification when the end document node were found

virtual void EndElement (System.String ns, System.String localName, System.String qName)

This method manage the notification when the end element node were found

Table 2.394. Parameters

ns	The namespace of the element
localName	The local name of the element
qName	The long name (qualify name) of the element

virtual void EndPrefixMapping (System.String prefix)

This method manage the event when an area of expecific URI prefix was ended.

Table 2.395. Parameters

prefix	The prefix that ends

virtual void Error (System.Xml.XmlException exception)

This method manage when an error exception ocurrs in the parsing process

Table 2.396. Parameters

exception	The exception thrown by the parser

virtual void FatalError (System.Xml.XmlException exception)

This method manage when a fatal error exception ocurrs in the parsing process

Table 2.397. Parameters

exception	The exception Thrown by the parser

virtual void IgnorableWhitespace (char[] chars, int start, int length)

This method manage the event when a ignorable whitespace node were found

Table 2.398. Parameters

chars	The array with the ignorable whitespaces
start	The array offset at the ignorable whitespace
length	The length of the whitespaces

virtual void ProcessingInstruction (System.String target, System.String data)

This method manage the event when a processing instruction were found

Table 2.399. Parameters

target	The processing instruction target
data	The processing instruction data

virtual XmlSource ResolveEntity (System.String publicId, System.String systemId)

Allow the application to resolve external entities.

Table 2.400. 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.

virtual void SetDocumentLocator (XmlSaxLocator locator)

This method is not supported, is include for compatibility

Table 2.401. Parameters

locator	A `XmlSaxLocator` object that can return the location of any events into the XML document

virtual void SkippedEntity (System.String name)

This method manage the event when a skipped entity were found

Table 2.402. Parameters

name	The name of the skipped entity

virtual void StartDocument ()

This method manage the event when a start document node were found

virtual void StartElement (System.String ns, System.String localName, System.String qName, XmlAttributes attributes)

This method manage the event when a start element node were found

Table 2.403. 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

virtual void StartPrefixMapping (System.String prefix, System.String uri)

This methods indicates the start of a prefix area in the XML document.

Table 2.404. Parameters

prefix	The prefix of the area
uri	The namespace uri of the prefix area

virtual void Warning (System.Xml.XmlException exception)

This method manage when a warning exception ocurrs in the parsing process

Table 2.405. Parameters

exception	The exception Thrown by the parser

Com::Objsys::Asn1::Runtime::XmlSaxEntityResolver interface Reference

- XmlSource ResolveEntity (System.String publicId, System.String systemId)

Detailed Description

Basic interface for resolving entities.

Definition at line 33 of file XmlSaxEntityResolver.cs

The Documentation for this struct was generated from the following file:

- XmlSaxEntityResolver.cs

XmlSource ResolveEntity (System.String publicId, System.String systemId)

Allow the application to resolve external entities.

Table 2.406. 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.

Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler interface Reference

- void Error (System.Xml.XmlException exception)
- void FatalError (System.Xml.XmlException exception)
- void Warning (System.Xml.XmlException exception)

Detailed Description

This interface will manage errors during the parsing of a XML document.

Definition at line 33 of file XmlSaxErrorHandler.cs

The Documentation for this struct was generated from the following file:

- XmlSaxErrorHandler.cs

void Error (System.Xml.XmlException exception)

This method manage an error exception ocurred during the parsing process.

Table 2.407. Parameters

exception	The exception thrown by the parser.

void FatalError (System.Xml.XmlException exception)

This method manage a fatal error exception ocurred during the parsing process.

Table 2.408. Parameters

exception	The exception thrown by the parser.

void Warning (System.Xml.XmlException exception)

This method manage a warning exception ocurred during the parsing process.

Table 2.409. Parameters

exception	The exception thrown by the parser.

Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler interface Reference

- 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)

Detailed Description

This interface will manage the Content events of a XML document.

Definition at line 33 of file XmlSaxLexicalHandler.cs

The Documentation for this struct was generated from the following file:

- XmlSaxLexicalHandler.cs

void Comment (char[] ch, int start, int length)

This method manage the notification when Characters elements were found.

Table 2.410. 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.

Table 2.411. 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.

Table 2.412. 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.

Table 2.413. Parameters

name	The name of the entity that is ending.

Com::Objsys::Asn1::Runtime::XmlSaxLocator interface Reference

- int GetColumnNumber ( )
- int GetLineNumber ( )
- System.String GetPublicId ( )
- System.String GetSystemId ( )

Detailed Description

This interface is created to emulate the SAX Locator interface behavior.

Definition at line 33 of file XmlSaxLocator.cs

The Documentation for this struct was generated from the following file:

- XmlSaxLocator.cs

int GetColumnNumber ()

This method return the column number where the current document event ends.

Returns: . The column number where the current document event ends.

int GetLineNumber ()

This method return the line number where the current document event ends.

Returns: . The line number where the current document event ends.

System.String GetPublicId ()

This method is not supported, it is included for compatibility.

Returns: . The saved public identifier.

System.String GetSystemId ()

This method is not supported, it is included for compatibility.

Returns: . The saved system identifier.

Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl class Reference

Private Attributes

- int columnNumber
- int lineNumber
- System.String publicId
- System.String systemId

- 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 ( )
- XmlSaxLocatorImpl (XmlSaxLocator locator)

Detailed Description

This class is created for emulates the SAX LocatorImpl behaviors.

Definition at line 33 of file XmlSaxLocatorImpl.cs

The Documentation for this struct was generated from the following file:

- XmlSaxLocatorImpl.cs

virtual int GetColumnNumber ()

Return the saved column number.

Returns: . The saved column number.

virtual int GetLineNumber ()

Return the saved line number.

Returns: . The saved line number.

virtual System.String GetPublicId ()

This method is not supported, it is included for compatibility. Return the saved public identifier.

Returns: . The saved public identifier.

virtual System.String GetSystemId ()

This method is not supported, it is included for compatibility. Return the saved system identifier.

Returns: . The saved system identifier.

virtual void SetColumnNumber (int columnNumber)

Set the column number for this locator.

Table 2.414. Parameters

columnNumber	The column number.

virtual void SetLineNumber (int lineNumber)

Set the line number for this locator.

Table 2.415. Parameters

lineNumber	The line number.

virtual void SetPublicId (System.String publicId)

This method is not supported, it is included for compatibility. Set the public identifier for this locator.

Table 2.416. Parameters

publicId	The new public identifier.

virtual void SetSystemId (System.String systemId)

This method is not supported, it is included for compatibility. Set the system identifier for this locator.

Table 2.417. Parameters

systemId	The new system identifier.

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.

Table 2.418. Parameters

locator	The current state of a locator.

Returns: . A new 'XmlSaxLocatorImpl' instance.

Com::Objsys::Asn1::Runtime::XmlSaxParser class Reference

Protected Attributes

- XmlSaxContentHandler callBackHandler
- XmlSaxEntityResolver entityResolver
- XmlSaxErrorHandler errorHandler
- XmlSaxLexicalHandler lexical
- XmlSaxLocatorImpl locator
- bool namespaceAllowed
- System.String parserFileName
- System.Xml.XmlTextReader reader
-

- virtual XmlSaxContentHandler GetContentHandler ( )
- virtual XmlSaxEntityResolver GetEntityResolver ( )
- virtual XmlSaxErrorHandler GetErrorHandler ( )
- virtual void Parse (System.IO.FileInfo filepath, XmlSaxContentHandler handler)
- virtual void Parse (System.String filepath, XmlSaxContentHandler handler)
- virtual void Parse (System.IO.Stream stream, XmlSaxContentHandler handler)
- virtual void Parse (System.IO.Stream stream, XmlSaxContentHandler handler, System.String URI)
- virtual void Parse (XmlSource source, XmlSaxContentHandler handler)
- virtual void Parse (System.IO.FileInfo filepath)
- virtual void Parse (System.String filepath)
- virtual void Parse (System.IO.Stream stream)
- virtual void Parse (System.IO.Stream stream, System.String URI)
- virtual void Parse (XmlSource source)
- virtual void SetContentHandler (XmlSaxContentHandler handler)
- virtual void SetDocumentHandler (XmlSaxContentHandler handler)

- virtual void SetEntityResolver (XmlSaxEntityResolver resolver)
- virtual void SetErrorHandler (XmlSaxErrorHandler handler)
- XmlSaxParser ( )

- static XmlSaxParser CloneInstance (XmlSaxParser instance)
- static XmlSaxParser NewInstance ( )

- void DoParsing ( )
- void UpdateLocatorData (XmlSaxLocatorImpl locator, System.Xml.XmlTextReader textReader)

Detailed Description

Emulates the SAX parsers behaviours.

Definition at line 33 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

Member Data Documentation

XmlSaxContentHandler callBackHandler

XmlSaxContentHandler variable manages the Content events of a XML document.

Definition at line 42 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

XmlSaxEntityResolver entityResolver

XmlSaxEntityResolver variable for resolving entities

Definition at line 51 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

XmlSaxErrorHandler errorHandler

XmlSaxErrorHandler variable manages errors during the parsing of a XML document

Definition at line 44 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

XmlSaxLexicalHandler lexical

XmlSaxLexicalHandler variable manages the Content events of a XML document

Definition at line 49 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

XmlSaxLocatorImpl locator

XmlSaxLocatorImpl variable to emulates the SAX LocatorImpl behaviors.

Definition at line 47 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

bool namespaceAllowed

Bool variable manages XML document namespace processing.

Definition at line 36 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

System.String parserFileName

String variable holds the XER or XML message file name

Definition at line 54 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

System.Xml.XmlTextReader reader

XmlTextReader variable manages XML document parsing.

Definition at line 39 of file XmlSaxParser.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParser.cs

virtual XmlSaxContentHandler GetContentHandler ()

Obtains the object that will handle all the content events.

Returns: . The object that handles the content events.

virtual XmlSaxEntityResolver GetEntityResolver ()

Returns the current entity resolver.

Returns: . The current entity resolver, or null if none has been registered.

virtual XmlSaxErrorHandler GetErrorHandler ()

Assigns the object that will handle all the error events.

Returns: . The object that handles the error events.

virtual void Parse (System.IO.FileInfo filepath, XmlSax-ContentHandler handler)

Parses the specified file and process the events over the specified handler.

Table 2.419. Parameters

filepath	The file to be used.
handler	The handler that manages the parser events.

virtual void Parse (System.String filepath, XmlSaxCon-tentHandler handler)

Parses the specified file path and process the events over the specified handler.

Table 2.420. Parameters

filepath	The path of the file to be used.
handler	The handler that manage the parser events.

virtual void Parse (System.IO.Stream stream, XmlSax-ContentHandler handler)

Parses the specified stream and process the events over the specified handler.

Table 2.421. Parameters

stream	The stream with the XML.

handler	The handler that manage the parser events.

virtual void Parse (System.IO.Stream stream, XmlSax-ContentHandler handler, System.String URI)

Parses the specified stream and process the events over the specified handler, and resolves the entities with the specified URI.

Table 2.422. Parameters

stream	The stream with the XML.
handler	The handler that manage the parser events.
URI	The namespace URI for resolve external etities.

virtual void Parse (XmlSource source, XmlSaxCon-tentHandler handler)

Parses the specified 'XmlSource' instance and process the events over the specified handler, and resolves the entities with the specified URI.

Table 2.423. Parameters

source	The 'XmlSource' that contains the XML.
handler	The handler that manages the parser events.

virtual void Parse (System.IO.FileInfo filepath)

Parses the specified file and process the events over previously specified handler.

Table 2.424. Parameters

filepath	The file with the XML.

virtual void Parse (System.String filepath)

Parses the specified file path and processes the events over previously specified handler.

Table 2.425. Parameters

filepath	The path of the file with the XML.

virtual void Parse (System.IO.Stream stream)

Parses the specified stream and process the events over previusly specified handler.

Table 2.426. Parameters

stream	The stream with the XML.

virtual void Parse (System.IO.Stream stream, System.String URI)

Parses the specified stream and processes the events over previously specified handler, and resolves the external entities with the specified URI.

Table 2.427. Parameters

stream	The stream with the XML.
URI	The namespace URI for resolve external etities.

virtual void Parse (XmlSource source)

Parses the specified 'XmlSource' and processes the events over the specified handler, and resolves the entities with the specified URI.

Table 2.428. Parameters

source	The 'XmlSource' instance with the XML.

virtual void SetContentHandler (XmlSaxContentHandler handler)

Assigns the object that will handle all the content events.

Table 2.429. Parameters

handler	The object that handles the content events.

virtual void SetDocumentHandler (XmlSaxContentHandler handler)

Assigns the object that will handle all the document events.

Table 2.430. Parameters

handler	The object that handles the content events.

virtual void SetEntityResolver (XmlSaxEntityResolver resolver)

Allows an application to register an entity resolver.

Table 2.431. Parameters

resolver	The entity resolver.

virtual void SetErrorHandler (XmlSaxErrorHandler handler)

Assigns the object that will handle all the error events.

Table 2.432. Parameters

handler	The object that handles the errors events.

XmlSaxParser ()

Public constructor for the class.

static XmlSaxParser CloneInstance (XmlSaxParser instance)

Create a clone instance of 'XmlSaxParser'.

Table 2.433. Parameters

instance	The `XmlSaxParser` instance to be cloned.

Returns: . A clone 'XmlSaxParser' instance.

static XmlSaxParser NewInstance ()

Returns a new instance of 'XmlSaxParser'.

Returns: . A new 'XmlSaxParser' instance.

void DoParsing ()

Emulates the behavior of a SAX parser, it realizes the callback events of the parser.

void UpdateLocatorData (XmlSaxLocatorImpl locator, System.Xml.XmlTextReader textReader)

Emulates the behaviour of a SAX LocatorImpl object.

Table 2.434. Parameters

locator	The 'XmlSaxLocatorImpl' instance to assing the document location.
textReader	The XML document instance to be used.

Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter class Reference

- 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)

Detailed Description

This class provides the base implementation for the management of XML documents parsing.

Definition at line 33 of file XmlSaxParserAdapter.cs

The Documentation for this struct was generated from the following file:

- XmlSaxParserAdapter.cs

virtual void Characters (char[] ch, int start, int length)

This method manage the notification when Characters element were found.

Table 2.435. 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

virtual void EndDocument ()

This method manage the notification when the end document node were found

virtual void EndElement (System.String namespaceURI, System.String localName, System.String qName)

This method manage the notification when the end element node were found

Table 2.436. Parameters

namespaceURI	The namespace URI of the element
localName	The local name of the element
qName	The long name (qualify name) of the element

virtual void EndPrefixMapping (System.String prefix)

This method manage the event when an area of expecific URI prefix was ended.

Table 2.437. Parameters

prefix	The prefix that ends.

virtual void IgnorableWhitespace (char[] ch, int start, int length)

This method manage the event when a ignorable whitespace node were found

Table 2.438. Parameters

ch	The array with the ignorable whitespaces

start	The index in the array with the ignorable whitespace
length	The length of the whitespaces

virtual void ProcessingInstruction (System.String target, System.String data)

This method manage the event when a processing instruction were found

Table 2.439. Parameters

target	The processing instruction target
data	The processing instruction data

virtual void SetDocumentLocator (XmlSaxLocator locator)

Receive an object for locating the origin of events into the XML document

Table 2.440. Parameters

locator	A `XmlSaxLocator` object that can return the location of any events into the XML document

virtual void SkippedEntity (System.String name)

This method manage the event when a skipped entity was found.

Table 2.441. Parameters

name	The name of the skipped entity.

virtual void StartDocument ()

This method manage the event when a start document node were found

virtual void StartElement (System.String namespaceURI, System.String localName, System.String qName, XmlAttributes qAtts)

This method manage the event when a start element node were found

Table 2.442. 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

virtual void StartPrefixMapping (System.String prefix, System.String uri)

This methods indicates the start of a prefix area in the XML document.

Table 2.443. Parameters

prefix	The prefix of the area.
uri	The namespace URI of the prefix area.

Com::Objsys::Asn1::Runtime::XmlSource class Reference

Private Attributes

- System.IO.Stream bytes
- System.IO.StreamReader characters
- System.String uri
-
-
-

- XmlSource ( )
- XmlSource (System.IO.Stream stream)
- XmlSource (System.IO.StreamReader reader)
- XmlSource (System.String source)

Detailed Description

This class is used to encapsulate a source of Xml code in an single class.

Definition at line 33 of file XmlSource.cs

The Documentation for this struct was generated from the following file:

- XmlSource.cs

XmlSource ()

Constructs an empty XmlSource instance.

XmlSource (System.IO.Stream stream)

Constructs a XmlSource instance with the specified source System.IO.Stream.

Table 2.444. Parameters

stream	The stream containing the document.

XmlSource (System.IO.StreamReader reader)

Constructs a XmlSource instance with the specified source System.IO.StreamReader.

Table 2.445. Parameters

reader	The reader containing the document.

XmlSource (System.String source)

Construct a XmlSource instance with the specified source Uri string.

Table 2.446. Parameters

source	The source containing the document.

Chapter 3. File Documentation

Asn1BerDecodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime
- struct System

Detailed Description

Definition in file Asn1BerDecodeBuffer.cs

Asn1BerDecodeContext.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerDecodeContext

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1BerDecodeContext.cs

Asn1BerEncodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1BerEncodeBuffer.cs

Asn1BerInputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerInputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1BerInputStream.cs

Asn1BerMessageDumpHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerMessageDumpHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1BerMessageDumpHandler.cs

Asn1BerOutputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1BerOutputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1BerOutputStream.cs

Asn1CerInputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1CerInputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1CerInputStream.cs

Asn1CerOutputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1CerOutputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1CerOutputStream.cs

Asn1DerDecodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1DerDecodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1DerDecodeBuffer.cs

Asn1DerEncodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1DerEncodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1DerEncodeBuffer.cs

Asn1DerInputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1DerInputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1DerInputStream.cs

Asn1MderDecodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1MderDecodeBuffer

  Asn1MderDecodeBuffer provides the source for an MDER decode.

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1MderDecodeBuffer.cs

Asn1MderOutputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1MderOutputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1MderOutputStream.cs

Asn1MderUnsupported.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1MderUnsupported

  <summary> Generally, the conditions under which this exception would be thrown should be detected at the time of ASN.1 compilation, so this exception both suggests a code generation error and an incompatibility between the compiled ASN.1 and the MDER.

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1MderUnsupported.cs

Asn1NotInSetException.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1NotInSetException

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1NotInSetException.cs

Asn1OerDecodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1OerDecodeBuffer

  This class handles the decoding of ASN.1 messages as specified in the Octet Encoding Rules (OER) as documented in the ITU-T X.696 standard.

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1OerDecodeBuffer.cs

Asn1OerDecoder.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1OerDecoder

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1OerDecoder.cs

Asn1PerBitField.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerBitField

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerBitField.cs

Asn1PerBitFieldList.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerBitFieldList

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerBitFieldList.cs

Asn1PerBitFieldPrinter.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerBitFieldPrinter

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerBitFieldPrinter.cs

Asn1PerDecodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerDecodeBuffer.cs

Asn1PerDecodeTraceHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerDecodeTraceHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerDecodeTraceHandler.cs

Asn1PerEncodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerEncodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerEncodeBuffer.cs

Asn1PerEncodeTraceHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerEncodeTraceHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerEncodeTraceHandler.cs

Asn1PerInputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerInputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerInputStream.cs

Asn1PerMessageBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerMessageBuffer.cs

Asn1PerOutputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerOutputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerOutputStream.cs

Asn1PerOutputStreamTraceHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerOutputStreamTraceHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerOutputStreamTraceHandler.cs

Asn1PerTraceHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerTraceHandler.cs

Asn1PerUtil.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1PerUtil

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1PerUtil.cs

Asn1SetDuplicateException.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1SetDuplicateException

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1SetDuplicateException.cs

Asn1TaggedEventHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1TaggedEventHandler.cs

Asn1TagMatchFailedException.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1TagMatchFailedException

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1TagMatchFailedException.cs

Asn1XerDecodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerDecodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerDecodeBuffer.cs

Asn1XerElemInfo.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerElemInfo

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerElemInfo.cs

Asn1XerEncodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerEncodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerEncodeBuffer.cs

Asn1XerEncoder.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerEncoder

- struct Com::Objsys::Asn1::Runtime::Asn1XerEncoder_Fields

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerEncoder.cs

Asn1XerOpenType.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerOpenType

Namespaces

- struct Com::Objsys::Asn1::Runtime
- struct System::Runtime::InteropServices

Detailed Description

Definition in file Asn1XerOpenType.cs

Asn1XerOutputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerOutputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerOutputStream.cs

Asn1XerSaxHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerSaxHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerSaxHandler.cs

Asn1XerUtil.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XerUtil

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XerUtil.cs

Asn1XmlEncodeBuffer.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XmlEncodeBuffer

Namespaces

- struct Com::Objsys::Asn1::Runtime
- struct System::Collections

Detailed Description

Definition in file Asn1XmlEncodeBuffer.cs

Asn1XmlOutputStream.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XmlOutputStream

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file Asn1XmlOutputStream.cs

Asn1XmlUtil.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::Asn1XmlUtil
- struct Com::Objsys::Asn1::Runtime::Asn1XmlUtil::CaptureElementTracking

Namespaces

- struct Com::Objsys::Asn1::Runtime
- struct System::Collections::Generic

Detailed Description

Definition in file Asn1XmlUtil.cs

XmlAttributes.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlAttributes::XmlAttribute
- struct Com::Objsys::Asn1::Runtime::XmlAttributes

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlAttributes.cs

XmlSaxContentHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxContentHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxContentHandler.cs

XmlSaxDefaultHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxDefaultHandler.cs

XmlSaxEntityResolver.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxEntityResolver

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxEntityResolver.cs

XmlSaxErrorHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxErrorHandler.cs

XmlSaxLexicalHandler.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxLexicalHandler

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxLexicalHandler.cs

XmlSaxLocator.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxLocator

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxLocator.cs

XmlSaxLocatorImpl.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxLocatorImpl

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxLocatorImpl.cs

XmlSaxParser.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxParser

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxParser.cs

XmlSaxParserAdapter.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSaxParserAdapter

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSaxParserAdapter.cs

XmlSource.cs File Reference

Classes

- struct Com::Objsys::Asn1::Runtime::XmlSource

Namespaces

- struct Com::Objsys::Asn1::Runtime

Detailed Description

Definition in file XmlSource.cs