ASN1C

ASN.1 Compiler
Version 7.1
C# BER/DER/PER/XER/XML
Reference Manual

Objective Systems, Inc. version 7.1 — January 2017

Copyright Notice

Author's Contact Information

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

Contents

Chapter 1

Class Documentation

1.1 Asn1BerDecodeBuffer Class Reference

Inherited by Asn1BerInputStream, and Asn1DerDecodeBuffer.

Public Member Functions

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

Static Public Member Functions

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

Protected Member Functions

- internal void MovePastEOC (bool saveData)

Properties

- virtual Asn1Tag LastTag `[get]`

1.1.1 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.

1.1.2 Constructor & Destructor Documentation

1.1.2.1 Asn1BerDecodeBuffer (byte[ ] *msgdata*)

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

Parameters

msgdata Byte array containing an encoded ASN.1 message.

1.1.2.2 Asn1BerDecodeBuffer (System.IO.Stream *istream*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

1.1.3 Member Function Documentation

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

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

Parameters

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

Returns

calculated length

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

This method decodes an enumerated value from the buffer.

Parameters

tag An Asn1Tag value for enumerated values that are tagged other than UNIVERSAL 10.
explicitTagging A flag that indicates the element is explicitly tagged.
implicitLength The length of the contents if implicitly tagged.

Returns

The decoded integer value.

1.1.3.3 int DecodeEnumValue (bool *explicitTagging*, int *implicitLength*)

This method decodes an enumerated value from the buffer.

Parameters

explicitTagging A flag that indicates the element is explicitly tagged.
implicitLength The length of the contents if implicitly tagged.

Returns

The decoded integer value.

1.1.3.4 virtual int DecodeLength () `[virtual]`

This method decodes a length value.

Returns

Decoded length value

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

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

Parameters

saveData True if data should be captured and returned

Returns

Reference to byte array containing component.

1.1.3.6 virtual byte [ ] DecodeOpenType () `[virtual]`

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

Returns

Reference to byte array containing component.

1.1.3.7 virtual void DecodeTag (Asn1Tag *tag*) `[virtual]`

This method decodes a tag value.

Parameters

tag Tag object to receive decoded tag fields.

Returns

status value (see Asn1Status.java)

1.1.3.8 virtual int DecodeTagAndLength (Asn1Tag *tag*) [virtual]

This method decodes a tag and length value.

Parameters

tag Tag object to receive decoded tag fields.

Returns

Decoded length value.

1.1.3.9 virtual bool MatchTag (Asn1Tag *tag*) [virtual]

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

Parameters

tag Tag value to be matched.

Returns

True if given tag matches tag at Decode cursor

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

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

Parameters

tag Tag value to be matched.
parsedTag Holder object to receive parsed tag value
parsedLen Holder object to receive parsed length value

Returns

True if given tag matches tag at Decode cursor

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

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

Parameters

tagClass Class value of tag to match
tagForm Form value of tag to match
tagIDCode ID code of tag to match
parsedTag Holder object to receive parsed tag value
parsedLen Holder object to receive parsed length value

Returns

True if given tag matches tag at Decode cursor

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

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

Parameters

saveData True if data should be captured

1.1.3.13 virtual void Parse (Asn1TaggedEventHandler *handler*) `[virtual]`

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

Parameters

handler Object implementing the Asn1EventHandler interface.

Returns

Status value

1.1.3.14 virtual Asn1Tag PeekTag () `[virtual]`

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

Returns

Parsed tag object value reference

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

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

Parameters

parsedTag Holder object to receive parsed tag value

1.1.3.16 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

1.1.4 Property Documentation

1.1.4.1 virtual Asn1Tag LastTag `[get]`

Gets the last tag parsed within this decode buffer object.

`Value:` Last parsed tag object reference

1.2 Asn1BerDecodeContext Class Reference

Public Member Functions

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

Protected Attributes

- internal int mDecBufByteCount
- internal Asn1BerDecodeBuffer mDecodeBuffer
- internal int mElemLength
- internal bool mExplicitTagging
- internal Asn1Tag mTagHolder

1.2.1 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.

1.2.2 Constructor & Destructor Documentation

1.2.2.1 Asn1BerDecodeContext (Asn1BerDecodeBuffer *decodeBuffer*, int *elemLength*)

The constructor initializes all internal working varaibles.

Parameters

decodeBuffer Reference to current Decode buffer method.

elemLength Length of the element being tracked.

1.2.3 Member Function Documentation

1.2.3.1 virtual bool Expired () `[virtual]`

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

Returns

True if at the end of the context block

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

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

Parameters

tag Tag object representing tag to be matched.

parsedLen Holder object to receive parsed length value

advance True if Decode cursor to be advanced.

Returns

True, if the tag is matched

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

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

Parameters

tagClass Class value of tag to match

tagForm Form value of tag to match

tagIDCode ID code of tag to match

parsedLen Holder object to receive parsed length value

advance True if Decode cursor to be advanced.

Returns

True, if the tag is matched

1.2.4 Member Data Documentation

1.2.4.1 internal int mDecBufByteCount [protected]

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

1.2.4.2 internal Asn1BerDecodeBuffer mDecodeBuffer [protected]

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

1.2.4.3 internal int mElemLength [protected]

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

1.2.4.4 internal bool mExplicitTagging [protected]

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

1.2.4.5 internal Asn1Tag mTagHolder [protected]

This variable holds the current parsed tag for matching operations.

1.3 Asn1BerEncodeBuffer Class Reference

Inherited by Asn1DerEncodeBuffer.

Public Member Functions

- Asn1BerEncodeBuffer (int sizeIncrement)
- Asn1BerEncodeBuffer ()
- virtual void BinDump ()
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- virtual int EncodeIdentifier (int ident)
- virtual int EncodeIntValue (long ivalue)
- virtual int EncodeLength (int len)
- virtual int EncodeTag (Asn1Tag tag)
- virtual int EncodeTagAndLength (short tagClass, short tagForm, int tagIDCode, int len)
- virtual int EncodeTagAndLength (Asn1Tag tag, int len)
- virtual int EncodeUnsignedBinaryNumber (long ivalue)
- virtual int TrimBitString (Asn1BitString bitstr)

Protected Member Functions

- internal override void CheckSize (int bytesRequired)

1.3.1 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.

1.3.2 Constructor & Destructor Documentation

1.3.2.1 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.

1.3.2.2 Asn1BerEncodeBuffer (int *sizeIncrement*)

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

Parameters

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

1.3.3 Member Function Documentation

1.3.3.1 virtual void BinDump () `[virtual]`

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

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

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

Parameters

outs StreamWriter where dump will be printed

varName Name of the Decoded ASN1 Type

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

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

Parameters

bytesRequired Number of required bytes.

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

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

Parameters

ident The identifier to be encoded.

Returns

Length of the encoded component in octets.

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

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

Parameters

ivalue Integer value to encode

Returns

Length of encoded component

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

This method encodes a length value.

Parameters

len The length to be encoded.

Returns

Length of encoded component

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

This method encodes a tag value.

Parameters

tag The tag to be encoded.

Returns

Length of component or negative status value

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

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

Parameters

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

len The length to be encoded.

Returns

status value (see Asn1Status.java)

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

This method encodes both a tag and length value.

Parameters

tag The tag to be encoded.

len The length to be encoded.

Returns

Length of encoded component

1.3.3.10 virtual int EncodeUnsignedBinaryNumber (long *ivalue*) `[virtual]`

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

Parameters

ivalue Integer value to encode

Returns

Length of encoded component

1.3.3.11 virtual int TrimBitString (Asn1BitString *bitstr*) `[virtual]`

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>

Reimplemented in Asn1DerEncodeBuffer.

1.4 Asn1BerInputStream Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer.

Inherited by Asn1CerInputStream.

Public Member Functions

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

1.4.1 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.

1.4.2 Constructor & Destructor Documentation

1.4.2.1 Asn1BerInputStream (System.IO.Stream *istream*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

1.4.3 Member Function Documentation

1.4.3.1 virtual int Available () `[virtual]`

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

Returns

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

Exceptions

System.SystemException if an I/O error occurs.

1.4.3.2 virtual void Close () `[virtual]`

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

Exceptions

System.SystemException if an I/O error occurs.

1.4.3.3 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.

1.4.3.4 virtual bool MarkSupported () `[virtual]`

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

Returns

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

1.4.3.5 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.

1.4.3.6 override long Skip (long *nbytes*)

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

Parameters

nbytes Number of bytes to skip

Exceptions

System.SystemException if an I/O error occurs.

Returns

Skipped number of bytes

1.5 Asn1BerMessageDumpHandler Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1TaggedEventHandler.

Public Member Functions

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

1.5.1 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.

1.5.2 Constructor & Destructor Documentation

1.5.2.1 Asn1BerMessageDumpHandler ()

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

1.5.2.2 Asn1BerMessageDumpHandler (System.IO.StreamWriter *outs*)

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

Parameters

outs Output stream for formatted data

1.5.3 Member Function Documentation

1.5.3.1 virtual void Contents (byte[ ] *data*) `[virtual]`

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

Parameters

data Array containing the encoded contents bytes

Implements Asn1TaggedEventHandler.

1.5.3.2 virtual void EndElement (Asn1Tag *tag*) `[virtual]`

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

Parameters

tag Array containing the encoded contents bytes

Implements Asn1TaggedEventHandler.

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

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

Parameters

tag Parsed tag value

len Parsed length value

tagLenBytes Array containing the encoded tag/length bytes

Implements Asn1TaggedEventHandler.

1.6 Asn1BerOutputStream Class Reference

Inherited by Asn1CerOutputStream.

Public Member Functions

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

1.6.1 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.

1.6.2 Constructor & Destructor Documentation

1.6.2.1 Asn1BerOutputStream (System.IO.Stream *os*)

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

Parameters

os The underlying System.IO.Stream object.

1.6.2.2 Asn1BerOutputStream (System.IO.Stream *os*, int *bufSize*)

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

Parameters

os The underlying System.IO.Stream object.

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

1.6.3 Member Function Documentation

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

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

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

Parameters

type The object to be written

explicitTagging Flag indicating explicit tagging should be done

Reimplemented in Asn1CerOutputStream.

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

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

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

Parameters

data Byte array containing data to encode.

numbits Number of bits to encode

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented in Asn1CerOutputStream.

1.6.3.3 virtual void EncodeBMPString (System.String *data*, bool *explicitTagging*, Asn1Tag *tag*) [virtual]

This method writes the given string as BMP string value.

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

Parameters

data String containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented in Asn1CerOutputStream.

1.6.3.4 virtual void EncodeCharString (System.String *data*, bool *explicitTagging*, Asn1Tag *tag*) [virtual]

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

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

Parameters

data The string object to be written

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented in Asn1CerOutputStream.

1.6.3.5 virtual void EncodeEOC () [virtual]

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

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

1.6.3.6 virtual void EncodeIdentifier (long *ident*) [virtual]

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

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

Parameters

ident The identifier to be encoded.

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

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

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

Parameters

data Integer value to encode.

encodeLen Flag indicating length determinant should be encoded before encoding integer value.

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

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

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

Parameters

len The length to be encoded.

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

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

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

Parameters

data Byte array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented in Asn1CerOutputStream.

1.6.3.10 virtual void EncodeTag (short *tagClass*, short *tagForm*, int *tagIDCode*) [virtual]

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

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

Parameters

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

1.6.3.11 virtual void EncodeTag (Asn1Tag *tag*) [virtual]

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

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

Parameters

tag The tag to be encoded.

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

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

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

Parameters

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

1.6.3.13 virtual void EncodeTagAndIndefLen (Asn1Tag *tag*) [virtual]

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

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

Parameters

tag The tag to be encoded.

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

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

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

Parameters

tag The tag to be encoded.

len The length to be encoded.

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

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

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

Parameters

data Array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented in Asn1CerOutputStream.

1.6.3.16 virtual void EncodeUnsignedBinaryNumber (long *data*) `[virtual]`

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

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

Parameters

data Integer value to encode.

1.7 Asn1CerInputStream Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1BerInputStream.

Public Member Functions

- Asn1CerInputStream (System.IO.Stream istream)

1.7.1 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.

1.7.2 Constructor & Destructor Documentation

1.7.2.1 Asn1CerInputStream (System.IO.Stream *istream*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

1.8 Asn1CerOutputStream Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1BerOutputStream.

Public Member Functions

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

1.8.1 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.

1.8.2 Constructor & Destructor Documentation

1.8.2.1 Asn1CerOutputStream (System.IO.Stream *os*)

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

Parameters

os The underlying System.IO.Stream object.

1.8.2.2 Asn1CerOutputStream (System.IO.Stream *os*, int *bufSize*)

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

Parameters

os The underlying System.IO.Stream object.

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

1.8.3 Member Function Documentation

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

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

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

Parameters

type The object to be written

explicitTagging Flag indicating explicit tagging should be done

Reimplemented from Asn1BerOutputStream.

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

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

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

Parameters

value Byte array containing data to encode.

numbits Number of bits to encode

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from Asn1BerOutputStream.

1.8.3.3 override void EncodeBMPString (System.String *value*, bool *explicitTagging*, Asn1Tag *tag*) [virtual]

This method writes the given string as BMP string value.

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

Parameters

value String containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from Asn1BerOutputStream.

1.8.3.4 override void EncodeCharString (System.String *value*, bool *explicitTagging*, Asn1Tag *tag*) [virtual]

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

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

Parameters

value The string object to be written

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from Asn1BerOutputStream.

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

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

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

Parameters

value Byte array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from Asn1BerOutputStream.

1.8.3.6 virtual void EncodeStringTag (int *nbytes*, short *tagClass*, short *tagForm*, int *tagIDCode*) [virtual]

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

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

Parameters

nbytes The number of bytes in string to be encoded.

tagClass The class of the tag to be encoded.

tagForm The form of the tag to be encoded.

tagIDCode The ID code of the tag to be encoded.

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

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

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

Parameters

nbytes The number of bytes in string to be encoded.

tag The tag to be encoded.

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

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

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

Parameters

value Array containing data to encode.

explicitTagging Flag indicating explicit tagging should be done

tag Universal tag to apply

Exceptions

Asn1Exception Thrown, if operation is failed.

Reimplemented from Asn1BerOutputStream.

1.9 Asn1DerDecodeBuffer Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1BerDecodeBuffer.

Inherited by Asn1DerInputStream.

Public Member Functions

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

1.9.1 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.

1.9.2 Constructor & Destructor Documentation

1.9.2.1 Asn1DerDecodeBuffer (byte[ ] *msgdata*)

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

Parameters

msgdata Byte array containing an encoded ASN.1 message.

1.9.2.2 Asn1DerDecodeBuffer (System.IO.Stream *istream*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

1.10 Asn1DerEncodeBuffer Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1BerEncodeBuffer.

Public Member Functions

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

1.10.1 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.

1.10.2 Constructor & Destructor Documentation

1.10.2.1 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.

1.10.2.2 Asn1DerEncodeBuffer (int *sizeIncrement*)

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

Parameters

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

1.10.3 Member Function Documentation

1.10.3.1 override int TrimBitString (Asn1BitString *bitstr*) `[virtual]`

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>

Reimplemented from Asn1BerEncodeBuffer.

1.11 Asn1DerInputStream Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1DerDecodeBuffer.

Public Member Functions

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

1.11.1 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.

1.11.2 Constructor & Destructor Documentation

1.11.2.1 Asn1DerInputStream (System.IO.Stream *istream*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

1.11.3 Member Function Documentation

1.11.3.1 virtual int Available () `[virtual]`

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

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

Returns

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

1.11.3.2 virtual void Close () `[virtual]`

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

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

1.11.3.3 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.

1.11.3.4 virtual bool MarkSupported () `[virtual]`

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

Returns

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

1.11.3.5 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.

1.11.3.6 override long Skip (long *nbytes*)

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

Parameters

nbytes Number of bytes to skip

Returns

Skipped number of bytes

1.12 Asn1NotInSetException Class Reference

Public Member Functions

- Asn1NotInSetException (Asn1BerDecodeBuffer buffer, Asn1Tag tag)

1.12.1 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.

1.12.2 Constructor & Destructor Documentation

1.12.2.1 Asn1NotInSetException (Asn1BerDecodeBuffer *buffer*, Asn1Tag *tag*)

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

Parameters

buffer BER decode buffer object reference

tag Tag value of element that is not in the set

1.13 Asn1PerBitField Class Reference

Public Member Functions

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

Properties

- virtual int BitCount `[get, set]`
- virtual int BitOffset `[get, set]`
- virtual System.String Name `[get]`

1.13.1 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.

1.13.2 Constructor & Destructor Documentation

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

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

Parameters

name Name of the bit field.

bitOffset Offset within buffer to the bit field

bitCount Number of bits in the bit field

1.13.3 Member Function Documentation

1.13.3.1 virtual void SetBitCountAndOffset (int *count*, int *offset*) `[virtual]`

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

Parameters

count Number of bits in the bit field

offset Offset within buffer to the bit field

1.13.4 Property Documentation

1.13.4.1 virtual int BitCount `[get, set]`

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

`Value:` Number of bits.

1.13.4.2 virtual int BitOffset `[get, set]`

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

`Value:` Offset of the bitfield

1.13.4.3 virtual System.String Name `[get]`

This method returns the name assigned to the bit field.

`Value:` Bitfield name

1.14 Asn1PerBitFieldList Class Reference

Public Member Functions

- virtual void AddElemName (System.String name, int arrayx)
- virtual System.Collections.IEnumerator Iterator ()
- virtual Asn1PerBitField NewBitField (System.String nameSuffix, int bitOffset, int bitCount)
- virtual void RemoveLastElemName ()
- virtual void Reset ()

Properties

- virtual int BitOffset `[set]`
- virtual Asn1PerBitField CurrBitField `[get]`

1.14.1 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.

1.14.2 Member Function Documentation

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

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

Parameters

name Name component to append to string

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

1.14.2.2 virtual System.Collections.IEnumerator Iterator () `[virtual]`

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

Returns

System.Collections.IEnumerator value of this list

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

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

Parameters

nameSuffix Suffix to add to fully qualified name for this field (for example, 'length')

bitOffset Offset to the start of this field in bits from the beginning of the encode buffer.

bitCount Number of bits in the field.

Returns

Created bit field

1.14.2.4 virtual void RemoveLastElemName () `[virtual]`

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

1.14.2.5 virtual void Reset () `[virtual]`

This method resets the object.

1.14.3 Property Documentation

1.14.3.1 virtual int BitOffset `[set]`

Set the current bit offset in the bit field.

`Value:` The bit offset

1.14.3.2 virtual Asn1PerBitField CurrBitField `[get]`

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

`Value:` Current bit field

1.15 Asn1PerBitFieldPrinter Class Reference

Public Member Functions

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

Protected Attributes

- internal int mBitMask
- internal int mByteIndex
- internal int mCurrOctet
- internal System.IO.Stream mEncodedMessage
- internal int mFmtBitCharIdx
- internal System.Text.StringBuilder mFormatBuffer
- internal Asn1PerMessageBuffer mPerMessageBuffer

1.15.1 Detailed Description

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

1.15.2 Constructor & Destructor Documentation

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

Constructor

Parameters

perMessageBuffer PER encode or decode message buffer

encodedMessage Input stream of encoded message

1.15.3 Member Function Documentation

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

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

Parameters

outs Print stream

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

1.15.4 Member Data Documentation

1.15.4.1 internal int mBitMask `[protected]`

This variable holds the mask for current bit

1.15.4.2 internal int mByteIndex [protected]

This variable holds the byte index

1.15.4.3 internal int mCurrOctet [protected]

This variable holds the current byte

1.15.4.4 internal System.IO.Stream mEncodedMessage [protected]

This variable holds the input stream

1.15.4.5 internal int mFmtBitCharIdx [protected]

This variable holds the index for formated information

1.15.4.6 internal System.Text.StringBuilder mFormatBuffer [protected]

Initial value:

```
new System.Text.StringBuilder()
```

This variable holds the formated information of current byte

1.15.4.7 internal Asn1PerMessageBuffer mPerMessageBuffer [protected]

This variable holds the PER encode or decode message buffer

1.16 Asn1PerDecodeBuffer Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer.

Inherited by Asn1PerInputStream.

Public Member Functions

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

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

Protected Attributes

- internal Asn1PerTraceHandler mTraceHandler

Properties

- virtual Asn1PerTraceHandler TraceHandler `[get]`

1.16.1 Detailed Description

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

1.16.2 Constructor & Destructor Documentation

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

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

Parameters

msgdata Byte array containing an encoded ASN.1 message.

aligned `true` for specifying PER aligned; otherwise `false` for unaligned encoding.

1.16.2.2 Asn1PerDecodeBuffer (System.IO.Stream *istream*, bool *aligned*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

aligned Boolean specifying PER aligned or unaligned encoding.

1.16.3 Member Function Documentation

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

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

Parameters

outs StreamWriter object to which output should be written

varName Name of top-level message object variable

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

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

Parameters

varName Name of top-level message object variable

1.16.3.3 override void ByteAlign ()

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

Implements Asn1PerMessageBuffer.

1.16.3.4 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.

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

This method decodes a single bit value.

Parameters

ident Bit field identifier name for tracing.

Returns

Boolean value of bit that was decoded.

1.16.3.6 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

Parameters

nbits Number of bits to Decode

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

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

Parameters

nbits Number of bits to Decode

ident Bit field identifier name for tracing.

Returns

Integer value containing decoded bits

1.16.3.8 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'.

Parameters

nbits Number of bits to Decode

Returns

Long integer value containing decoded bits

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

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

Returns

Long integer value containing decoded bits

Parameters

nbits Number of bits to Decode

ident Bit field identifier name for tracing.

1.16.3.10 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'.

Parameters

data Octet array for decoded data

offset Starting byte offset into array

nbits Number of bits to Decode

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

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

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

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.

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

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

Parameters

data Octet array for decoded data

offset Starting byte offset into array

nbits Number of bits to Decode

ident Bit field identifier name for tracing.

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

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

Parameters

nchars Number of characters

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

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

sbuf String buffer to receive decoded result

1.16.3.14 virtual long DecodeConsWholeNumber (long *rangeValue*) [virtual]

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

Parameters

rangeValue lower - upper + 1

Returns

Decoded adjusted value = value - lower range endpoint value

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

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

Parameters

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

ident Tracing identifier

Returns

Decoded adjusted value = value - lower range endpoint value

1.16.3.16 virtual long DecodeExtLength () `[virtual]`

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

Returns

Bit offset to next element in buffer

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

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

Returns

Decoded long integer value

Parameters

nocts Number of octets to Decode

signExtend Sign extend resulting value

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

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

Returns

Decoded long integer value

Parameters

nocts Number of octets to Decode

signExtend Sign extend resulting value

ident Tracing identifier

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

This method decodes a constrained length determinant value.

Parameters

lower Lower bound (inclusive) of length value range

upper Upper bound (inclusive) of length value range

Returns

Decoded length value

1.16.3.20 virtual long DecodeLength () [virtual]

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.

1.16.3.21 virtual int DecodeSmallLength () [virtual]

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

1.16.3.22 virtual int DecodeSmallNonNegWholeNumber () [virtual]

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

Returns

Decoded int value

1.16.3.23 virtual long DecodeUnconsLength () [virtual]

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.

1.16.3.24 virtual bool IsAligned () [virtual]

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

Returns

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

Implements Asn1PerMessageBuffer.

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

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

Parameters

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

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

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

Parameters

buffer Existing message buffer object

msgdata Byte array containing message data

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

Returns

Asn1PerDecodeBuffer to read data from the given byte array

1.16.3.27 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.

1.16.3.28 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.

1.16.4 Member Data Documentation

1.16.4.1 internal Asn1PerTraceHandler mTraceHandler [protected]

Variable holds the PER message trace handler

1.16.5 Property Documentation

1.16.5.1 virtual Asn1PerTraceHandler TraceHandler [get]

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

Value: Asn1PerTraceHandler object

Implements Asn1PerMessageBuffer.

1.17 Asn1PerDecodeTraceHandler Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler.

Public Member Functions

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

1.17.1 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.

1.17.2 Constructor & Destructor Documentation

1.17.2.1 Asn1PerDecodeTraceHandler (Asn1PerDecodeBuffer *messageBuffer*)

This constructor initializes the internal trace handler member variables.

Parameters

messageBuffer PER decode message buffer object reference

1.17.3 Member Function Documentation

1.17.3.1 override void Enable () `[virtual]`

This method is used to turn PER bit tracing on

Implements Asn1PerTraceHandler.

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

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

Parameters

outs Print stream to which output is to be written.

varName Name of the object variable being printed.

Implements Asn1PerTraceHandler.

1.17.3.3 override void Reset () `[virtual]`

This method resets the trace bit field list.

Implements Asn1PerTraceHandler.

1.18 Asn1PerEncodeBuffer Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1PerMessageBuffer.

Public Member Functions

- Asn1PerEncodeBuffer (bool aligned, int sizeIncrement)
- Asn1PerEncodeBuffer (bool aligned)
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- override void ByteAlign ()
- override void EncodeBit (bool value)
- virtual void EncodeBit (bool value, System.String ident)
- override void EncodeBits (byte[ ] value, int offset, int nbits)
- override void EncodeBits (byte[ ] value, int offset, int bitOffset, int nbits)
- virtual void EncodeBits (byte[ ] value, int offset, int bitOffset, int nbits, System.String ident)
- 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)
- virtual void EncodeConsWholeNumber (long adjustedValue, long rangeValue, System.String ident)
- virtual void EncodeInt (long value, bool encodeLen, bool signExtend)
- virtual void EncodeInt (long value, bool encodeLen, bool signExtend, System.String ident)
- virtual void EncodeInt (long value, int nbits)
- virtual void EncodeInt (long value, int nbits, System.String ident)
- virtual void EncodeLength (long value, long lower, long upper)
- virtual long EncodeLength (long value)
- virtual void EncodeLengthEOM (long value)
- virtual void EncodeOctetString (byte[ ] value, int offset, int nbytes)
- virtual void EncodeOIDLengthAndValue (int[ ] value)
- virtual void EncodeOpenType (Asn1PerEncodeBuffer buffer, System.String elemName)
- virtual void EncodeOpenType (byte[ ] value, int offset, int nbytes)
- virtual void EncodeRelOIDLengthAndValue (int[ ] value)
- virtual void 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)

Protected Attributes

- internal Asn1PerTraceHandler mTraceHandler

Properties

- virtual Asn1PerTraceHandler TraceHandler `[get]`

1.18.1 Detailed Description

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

1.18.2 Constructor & Destructor Documentation

1.18.2.1 Asn1PerEncodeBuffer (bool *aligned*)

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

Parameters

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

1.18.2.2 Asn1PerEncodeBuffer (bool *aligned*, int *sizeIncrement*)

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

Parameters

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

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

1.18.3 Member Function Documentation

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

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

Parameters

outs StreamWriter object to which output should be written

varName Name of top-level message object variable

1.18.3.2 override void ByteAlign ()

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

Implements Asn1PerMessageBuffer.

1.18.3.3 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");

Parameters

value Boolean value of bit to be encoded.

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

This method encodes a single bit value.

Parameters

value Boolean value of bit to be encoded.

ident Bit field identifier name for tracing.

1.18.3.5 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'.

Parameters

value Octet array containing bits to be encoded

offset Starting byte offset in value

nbits Number of bits to encode

1.18.3.6 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");

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

This method encodes bit values from an array of octets.

Parameters

value Octet array containing bits to be encoded

offset Starting byte offset in value

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.

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

This method encodes bit values from an array of octets.

Parameters

value Octet array containing bits to be encoded

offset Starting byte offset in value

nbits Number of bits to encode

ident Bit field identifier name for tracing.

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

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

Parameters

value String containing characters to encode

nchars Number of characters from string to encode

offset Offset to first char in string to encode

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing permitted alphabet constraint character set (optional)

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

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

Parameters

adjustedValue Adjusted value to be encoded = value - lower range endpoint value

rangeValue lower - upper + 1

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

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

Parameters

adjustedValue Adjusted value to be encoded = value - lower range endpoint value

rangeValue lower - upper + 1

ident Tracing identifier

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

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

Parameters

value Integer value to be encoded

encodeLen Flag indicating length determinant should be encoded before encoding integer value.

signExtend Flag indicating if sign extension should be performed.

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

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

Parameters

value Integer value to be encoded

encodeLen Flag indicating length determinant should be encoded before encoding integer value.

signExtend Flag indicating if sign extension should be performed.

ident Tracing identifier

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

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

Parameters

value Integer containing bits to be encoded

nbits Number of bits to encode

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

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

Parameters

value Integer containing bits to be encoded

nbits Number of bits to encode

ident Tracing identifier

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

This method encodes a constrained length determinant value.

Parameters

value Length value to be encoded

lower Lower bound (inclusive) of length value range

upper Upper bound (inclusive) of length value range

1.18.3.17 virtual long EncodeLength (long *value*) [virtual]

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

Parameters

value Length value to be encoded

Returns

Value that was actually encoded. This may be less than the value that was passed in if fragementation was done (i.e the value was >= 16k).

1.18.3.18 virtual void EncodeLengthEOM (long *value*) [virtual]

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

Parameters

value Original length value that was encoded.

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

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

Parameters

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

offset Starting offset in byte array value

nbytes Number of bytes to encode

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

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

Parameters

value Integer array containing arcs to encode

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

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

Parameters

buffer PER encode buffer containing encoded message component.

elemName Name of element being encoded.

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

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

Parameters

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

offset Starting offset in byte array value

nbytes Number of bytes to encode

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

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

Parameters

value Integer array containing arcs to encode

1.18.3.24 virtual void EncodeSmallLength (int *value*) [virtual]

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

Parameters

value Value to be encoded

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

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

Parameters

value Value to be encoded

1.18.3.26 virtual long EncodeUnconsLength (long *value*) [virtual]

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

Parameters

value Length value to be encoded

Returns

Value that was actually encoded. This may be less than the value that was passed in if fragementation was done (i.e the value was >= 16k).

1.18.3.27 override void HexDump ()

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

1.18.3.28 virtual bool IsAligned () [virtual]

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

Returns

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

Implements Asn1PerMessageBuffer.

1.18.3.29 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.

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

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

Parameters

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

1.18.3.31 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.

1.18.3.32 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.

1.18.4 Member Data Documentation

1.18.4.1 internal Asn1PerTraceHandler mTraceHandler [protected]

Variable holds the PER message trace handler

1.18.5 Property Documentation

1.18.5.1 virtual Asn1PerTraceHandler TraceHandler [get]

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

Value: Asn1PerTraceHandler object

Implements Asn1PerMessageBuffer.

1.19 Asn1PerEncodeTraceHandler Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler.

Public Member Functions

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

1.19.1 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.

1.19.2 Constructor & Destructor Documentation

1.19.2.1 Asn1PerEncodeTraceHandler (Asn1PerEncodeBuffer *messageBuffer*)

This constructor initializes internal trace handler member variables.

Parameters

messageBuffer PER encode message buffer object reference

1.19.3 Member Function Documentation

1.19.3.1 override void Enable () `[virtual]`

This method is used to turn PER bit tracing on

Implements Asn1PerTraceHandler.

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

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

Parameters

outs Print stream to which output is to be written.

varName Name of the object variable being printed.

Implements Asn1PerTraceHandler.

1.19.3.3 override void Reset () `[virtual]`

This method resets the trace bit field list.

Implements Asn1PerTraceHandler.

1.20 Asn1PerInputStream Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1PerDecodeBuffer.

Public Member Functions

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

1.20.1 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.

1.20.2 Constructor & Destructor Documentation

1.20.2.1 Asn1PerInputStream (System.IO.Stream *istream*, bool *aligned*)

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

Parameters

istream Input stream containing an encoded ASN.1 message.

aligned Boolean specifying PER aligned or unaligned encoding.

1.20.3 Member Function Documentation

1.20.3.1 virtual int Available () `[virtual]`

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

Returns

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

Exceptions

System.SystemException if an I/O error occurs.

1.20.3.2 virtual void Close () `[virtual]`

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

Exceptions

System.SystemException if an I/O error occurs.

1.20.3.3 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.

1.20.3.4 virtual bool MarkSupported () `[virtual]`

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

Returns

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

1.20.3.5 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.

1.20.3.6 override long Skip (long *nbytes*)

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

Parameters

nbytes Number of bytes to skip

Exceptions

System.SystemException if an I/O error occurs.

Returns

Skipped number of bytes

1.21 Asn1PerMessageBuffer Interface Reference

Inherited by Asn1PerDecodeBuffer, and Asn1PerEncodeBuffer.

Public Member Functions

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

Properties

- int MsgBitCnt `[get]`
- Asn1PerTraceHandler TraceHandler `[get]`

1.21.1 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.

1.21.2 Member Function Documentation

1.21.2.1 void ByteAlign ()

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

Implemented in Asn1PerDecodeBuffer, and Asn1PerEncodeBuffer.

1.21.2.2 System.IO.Stream GetInputStream ()

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

Returns

Stream containing message

1.21.2.3 bool IsAligned ()

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

Returns

`true` is aligned; otherwise `false`

Implemented in Asn1PerDecodeBuffer, and Asn1PerEncodeBuffer.

1.21.3 Property Documentation

1.21.3.1 int MsgBitCnt [get]

Gets the number of bits in the PER message.

Value: Count of bits in message

1.21.3.2 Asn1PerTraceHandler TraceHandler [get]

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

Value: Asn1PerTraceHandler object

Implemented in Asn1PerDecodeBuffer, and Asn1PerEncodeBuffer.

1.22 Asn1PerOutputStream Class Reference

Public Member Functions

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

Protected Attributes

- internal Asn1PerOutputStreamTraceHandler mTraceHandler

Properties

- virtual bool Aligned `[get]`
- virtual Asn1PerTraceHandler TraceHandler `[get]`

1.22.1 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.

1.22.2 Constructor & Destructor Documentation

1.22.2.1 Asn1PerOutputStream (System.IO.Stream *os*, bool *aligned*)

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

Parameters

os The underlying System.IO.Stream object.

aligned Indicates whether PER aligned or unaligned encoding should be done.

1.22.2.2 Asn1PerOutputStream (System.IO.Stream *os*, int *bufSize*, bool *aligned*)

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

Parameters

os The underlying System.IO.Stream object.

bufSize The buffer size.

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

1.22.3 Member Function Documentation

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

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

Parameters

buffer Buffer into which captured bytes are to be stored

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

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

Parameters

outs StreamWriter object to which output should be written

varName Name of top-level message object variable

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

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

Parameters

varName Name of top-level message object variable

1.22.3.4 virtual void ByteAlign () `[virtual]`

This methods byte-aligns the buffer.

1.22.3.5 override void Close ()

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

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

This method encodes a single bit value.

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

Parameters

value Boolean value of bit to be encoded.

ident Bit field identifier name for tracing.

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes a single bit value.

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

Parameters

value Boolean value of bit to be encoded.

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes bit values from an array of octets.

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

Parameters

value Octet array containing bits to be encoded

offset Starting byte offset in value

nbits Number of bits to encode

ident Bit field identifier name for tracing.

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes bit values from an array of octets.

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

Parameters

value Octet array containing bits to be encoded

offset Starting byte offset in value

nbits Number of bits to encode

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Octet containing bits to be encoded

nbits Number of bits to encode

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value String containing characters to encode

nchars Number of characters from string to encode

offset Offset to first char in string to encode

abpc Number of bits per character (aligned)

ubpc Number of bits per character (unaligned)

charSet Object representing permitted alphabet constraint character set (optional)

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

adjustedValue Adjusted value to be encoded = value - lower range endpoint value

rangeValue lower - upper + 1

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

adjustedValue Adjusted value to be encoded = value - lower range endpoint value

rangeValue lower - upper + 1

ident Bit field identifier name for tracing.

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Integer value to be encoded

encodeLen Flag indicating length determinant should be encoded before encoding integer value.

signExtend Flag indicating if sign extension should be performed.

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Integer value to be encoded

encodeLen Flag indicating length determinant should be encoded before encoding integer value.

signExtend Flag indicating if sign extension should be performed.

ident Bit field identifier name for tracing.

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Integer containing bits to be encoded

nbits Number of bits to encode

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Integer containing bits to be encoded

nbits Number of bits to encode

ident Bit field identifier name for tracing.

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

This method encodes a constrained length determinant value.

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

Parameters

value Length value to be encoded

lower Lower bound (inclusive) of length value range

upper Upper bound (inclusive) of length value range

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

1.22.3.19 virtual long EncodeLength (long *value*) `[virtual]`

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

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

Parameters

value Length value to be encoded

Returns

Value that was actually encoded. This may be less than the value that was passed in if fragementation was done (i.e the value was >= 16k).

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

1.22.3.20 virtual void EncodeLengthEOM (long *value*) `[virtual]`

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

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

Parameters

value Original length value that was encoded.

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

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

offset Starting offset in byte array value

nbytes Number of bytes to encode

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Integer array containing arcs to encode

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

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

offset Starting offset in byte array value

nbytes Number of bytes to encode

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Integer array containing arcs to encode

Exceptions

Asn1Exception Any exception thrown by the underlying Asn1PerEncodeBuffer.

1.22.3.25 virtual void EncodeSmallLength (int *value*) `[virtual]`

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.

Parameters

value Value to be encoded

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

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

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

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

Parameters

value Value to be encoded

Exceptions

Asn1InvalidArgException Any exception thrown by the underlying Asn1PerEncodeBuffer.

1.22.3.27 override void Flush ()

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

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

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

Parameters

buffer Buffer in which captured bytes stored

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

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

Parameters

b the data.

off The offset in array at which to begin write.

len The number of bytes to write.

Exceptions

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

1.22.3.30 override void Write (byte[ ] *b*)

Writes `b.length` bytes from the specified byte array to this output stream. The general contract for `write(b)` is that it should have exactly the same effect as the call `Write(b, 0, b.length)`.

Parameters

b the binary data.

Exceptions

System.SystemException if an I/O error occurs.

1.22.3.31 override void WriteByte (byte *b*)

Writes the specified byte to this output stream.

Parameters

b the byte.

Exceptions

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

1.22.3.32 override void WriteByte (int *b*)

Writes the specified byte to this output stream.

Parameters

b the byte.

Exceptions

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

1.22.4 Member Data Documentation

1.22.4.1 internal Asn1PerOutputStreamTraceHandler mTraceHandler [protected]

Variable holds the PER message trace handler

1.22.5 Property Documentation

1.22.5.1 virtual bool Aligned [get]

Gets PER aligned encoding has been specified.

Value: `true` is aligned; otherwise `false`

1.22.5.2 virtual Asn1PerTraceHandler TraceHandler [get]

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

Value: Asn1PerTraceHandler object

1.23 Asn1PerOutputStreamTraceHandler Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1PerTraceHandler.

Public Member Functions

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

1.23.1 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.

1.23.2 Constructor & Destructor Documentation

1.23.2.1 Asn1PerOutputStreamTraceHandler (Asn1PerOutputStream *outs*)

This constructor initializes the internal trace handler member variables.

Parameters

outs PER message stream object reference

1.23.3 Member Function Documentation

1.23.3.1 override void Enable () `[virtual]`

This method is used to turn PER bit tracing on

Implements Asn1PerTraceHandler.

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

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

Parameters

outs Print stream to which output is to be written.

varName Name of the object variable being printed.

Implements Asn1PerTraceHandler.

1.23.3.3 override void Reset () `[virtual]`

This method does nothing here.

Implements Asn1PerTraceHandler.

1.23.3.4 virtual void ResetTrace () [virtual]

This method resets the trace bit field list.

1.24 Asn1PerTraceHandler Class Reference

Inherited by Asn1PerDecodeTraceHandler, Asn1PerEncodeTraceHandler, and Asn1PerOutputStreamTraceHandler.

Public Member Functions

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

Protected Member Functions

- internal Asn1PerTraceHandler (Asn1PerMessageBuffer messageBuffer)

Protected Attributes

- internal Asn1PerBitFieldList mBitFieldList

Properties

- virtual Asn1PerBitFieldList BitFieldList `[get]`

1.24.1 Detailed Description

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

1.24.2 Constructor & Destructor Documentation

1.24.2.1 internal Asn1PerTraceHandler (Asn1PerMessageBuffer *messageBuffer*) `[protected]`

This constructor initializes internal trace handler member variables.

Parameters

messageBuffer PER message buffer object reference

1.24.3 Member Function Documentation

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

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

Parameters

name Name component to append to string

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

1.24.3.2 abstract void Enable () [pure virtual]

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

Implemented in Asn1PerEncodeTraceHandler, Asn1PerDecodeTraceHandler, and Asn1PerOutputStreamTraceHandler.

1.24.3.3 virtual void NewBitField (System.String *name*, int *bitCount*) [virtual]

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

Parameters

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

bitCount Number of bits in the bit field.

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

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

Parameters

outs Print stream to which output is to be written.

varName Name of the object variable being printed.

Implemented in Asn1PerEncodeTraceHandler, Asn1PerDecodeTraceHandler, and Asn1PerOutputStreamTraceHandler.

1.24.3.5 virtual void RemoveLastElemName () [virtual]

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.

1.24.3.6 abstract void Reset () [pure virtual]

This method resets the trace bit field list.

Implemented in Asn1PerEncodeTraceHandler, Asn1PerDecodeTraceHandler, and Asn1PerOutputStreamTraceHandler.

1.24.3.7 virtual void SetBitCount () [virtual]

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

1.24.3.8 virtual void SetBitOffset () [virtual]

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

1.24.4 Member Data Documentation

1.24.4.1 internal Asn1PerBitFieldList mBitFieldList [protected]

Variable holds the bit field list

1.24.5 Property Documentation

1.24.5.1 virtual Asn1PerBitFieldList BitFieldList [get]

Gets a reference to the bit field list

Value: bit field list

1.25 Asn1PerUtil Class Reference

Static Public Member Functions

- static int GetMsgBitCnt (int byteCount, int bitOffset)

1.25.1 Detailed Description

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

1.25.2 Member Function Documentation

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

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

Parameters

byteCount Number of full bytes in message

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

Returns

Count of bits in encoded message

1.26 Asn1SetDuplicateException Class Reference

Public Member Functions

- Asn1SetDuplicateException (Asn1BerDecodeBuffer buffer, Asn1Tag tag)

1.26.1 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..

1.26.2 Constructor & Destructor Documentation

1.26.2.1 Asn1SetDuplicateException (Asn1BerDecodeBuffer *buffer*, Asn1Tag *tag*)

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

Parameters

buffer BER decode buffer object reference

tag Tag value of duplicate element

1.27 Asn1TaggedEventHandler Interface Reference

Inherited by Asn1BerMessageDumpHandler.

Public Member Functions

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

1.27.1 Detailed Description

This interface defines the methods that must be implemented to define

a SAX-like event handler. These methods are invoked from within the generated C# decode logic when significant events occur during the parsing of an ASN.1 message.

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.

1.27.2 Member Function Documentation

1.27.2.1 void Contents (byte[ ] *data*)

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

Parameters

data Array containing encoded contents bytes.

Implemented in Asn1BerMessageDumpHandler.

1.27.2.2 void EndElement (Asn1Tag *tag*)

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

Parameters

tag Parsed tag value.

Implemented in Asn1BerMessageDumpHandler.

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

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

Parameters

tag Parsed tag value.

len Parsed length value

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

Implemented in Asn1BerMessageDumpHandler.

1.28 Asn1TagMatchFailedException Class Reference

Public Member Functions

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

1.28.1 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..

1.28.2 Constructor & Destructor Documentation

1.28.2.1 Asn1TagMatchFailedException (Asn1BerDecodeBuffer *buffer*, Asn1Tag *expectedTag*, Asn1Tag *parsedTag*)

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

Parameters

buffer BER decode buffer object reference

expectedTag Expected tag value

parsedTag Parsed tag value

1.28.2.2 Asn1TagMatchFailedException (Asn1BerDecodeBuffer *buffer*, Asn1Tag *expectedTag*)

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

Parameters

buffer BER decode buffer object reference

expectedTag Expected tag value

1.29 Asn1XerDecodeBuffer Class Reference

Public Member Functions

- Asn1XerDecodeBuffer (System.String source)

Protected Attributes

- internal XmlSource mInputSource

Properties

- virtual XmlSource InputSource `[get]`

1.29.1 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.

1.29.2 Constructor & Destructor Documentation

1.29.2.1 Asn1XerDecodeBuffer (System.String *source*)

This constructor creates an XER decode buffer.

Parameters

source The source containing the XML document.

1.29.3 Member Data Documentation

1.29.3.1 internal XmlSource mInputSource `[protected]`

Variable holds the SAX input source object.

1.29.4 Property Documentation

1.29.4.1 virtual XmlSource InputSource `[get]`

Gets the SAX input source object.

`Value:` SAX input source object

1.30 Asn1XerElemInfo Class Reference

Public Member Functions

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

Properties

- virtual int ID `[get]`
- virtual bool Optional `[get]`

1.30.1 Detailed Description

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

1.30.2 Constructor & Destructor Documentation

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

This constructor creates the element object.

Parameters

names first element names

optional true if component having given names is optional

id identifier

1.30.3 Member Function Documentation

1.30.3.1 bool Matches (System.String *name*)

Determines whether this object matches the given element name.

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`.

1.30.4 Property Documentation

1.30.4.1 virtual int ID `[get]`

Returns the ID value for the element.

`Value:` Identifier value

1.30.4.2 virtual bool Optional `[get]`

Determines whether the this element is optional

`Value:` `true` is optional; otherwise `false`

1.31 Asn1XerEncodeBuffer Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1XerEncoder.

Public Member Functions

- Asn1XerEncodeBuffer (bool canonical, int sizeIncrement)
- Asn1XerEncodeBuffer (bool canonical)
- Asn1XerEncodeBuffer ()
- override void BinDump (System.IO.StreamWriter outs, System.String varName)
- virtual void Copy (System.String data)
- virtual void 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 ()

Properties

- virtual bool Canonical `[set]`
- virtual int State `[get, set]`

1.31.1 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.

1.31.2 Constructor & Destructor Documentation

1.31.2.1 Asn1XerEncodeBuffer ()

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

1.31.2.2 Asn1XerEncodeBuffer (bool *canonical*)

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

Parameters

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

1.31.2.3 Asn1XerEncodeBuffer (bool *canonical*, int *sizeIncrement*)

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

Parameters

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

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

1.31.3 Member Function Documentation

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

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

Parameters

outs Output will be written to this stream

varName Name of the Decoded ASN1 Type

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

This method copies a character string to the encode buffer.

Parameters

data The string value to copy

1.31.3.3 virtual void DecrLevel () `[virtual]`

This method decrements the element nesting level counter.

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

This method encodes XML binary string data

Parameters

bits Bit String to encode

nbits Number of bits to encode

1.31.3.5 virtual void EncodeByte (byte *data*) [virtual]

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

Parameters

data The byte value to copy

1.31.3.6 virtual void EncodeData (System.String *data*) [virtual]

This method encodes XML string data

Parameters

data String value to encode

1.31.3.7 virtual void EncodeEmptyElement (System.String *elemName*) [virtual]

This method encodes an XML empty element tag

Parameters

elemName The name of element.

Implements Asn1XerEncoder.

1.31.3.8 virtual void EncodeEndDocument () [virtual]

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

1.31.3.9 virtual void EncodeEndElement (System.String *elemName*) [virtual]

This method encodes an XML end element tag

Parameters

elemName The name of element.

Implements Asn1XerEncoder.

1.31.3.10 virtual void EncodeHexStrValue (byte[ ] *data*) [virtual]

This method encodes XML hexadecimal string data

Parameters

data Data to encode

1.31.3.11 virtual void EncodeNamedValue (System.String *valueName*, System.String *elemName*) [virtual]

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

Parameters

elemName The name of element.

valueName named value.

Implements Asn1XerEncoder.

1.31.3.12 virtual void EncodeNamedValueElement (System.String *elemName*) [virtual]

This method encodes an XML named value element tag

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

1.31.3.13 virtual void EncodeRealValue (double *data*, System.String *elemName*) [virtual]

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

Parameters

data The value to be encoded.

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

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements Asn1XerEncoder.

1.31.3.14 virtual void EncodeStartDocument () [virtual]

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

1.31.3.15 virtual void EncodeStartElement (System.String *elemName*) `[virtual]`

This method encodes an XML start element tag

Parameters

elemName The name of element.

Implements Asn1XerEncoder.

1.31.3.16 virtual void IncrLevel () `[virtual]`

This method increments the element nesting level counter.

1.31.3.17 virtual void Indent () `[virtual]`

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

1.31.4 Property Documentation

1.31.4.1 virtual bool Canonical `[set]`

Sets the canonical encoding rule.

`Value:` `true` if canonical encoding; otherwise `false`.

1.31.4.2 virtual int State `[get, set]`

Sets the state of the buffer.

`Value:` buffer stat

Implements Asn1XerEncoder.

1.32 Asn1XerEncoder Interface Reference

Inherited by Asn1XerEncodeBuffer, and Asn1XerOutputStream.

Public Member Functions

- 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)

Properties

- int State `[get, set]`

1.32.1 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.

1.32.2 Member Function Documentation

1.32.2.1 void EncodeEmptyElement (System.String *elemName*)

This method encodes an XML empty element tag.

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

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in Asn1XerEncodeBuffer, and Asn1XerOutputStream.

1.32.2.2 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.

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in Asn1XerEncodeBuffer, and Asn1XerOutputStream.

1.32.2.3 void EncodeNamedValue (System.String *valueName*, System.String *elemName*)

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

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

Parameters

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

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in Asn1XerEncodeBuffer, and Asn1XerOutputStream.

1.32.2.4 void EncodeRealValue (double *valueName*, System.String *elemName*)

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

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

Parameters

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

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in Asn1XerEncodeBuffer, and Asn1XerOutputStream.

1.32.2.5 void EncodeStartElement (System.String *elemName*)

This method encodes an XML start element tag.

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

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implemented in Asn1XerEncodeBuffer, and Asn1XerOutputStream.

1.32.3 Property Documentation

1.32.3.1 int State `[get, set]`

This method gets and sets the state of the buffer.

`Value:` Buffer Stat

Implemented in Asn1XerEncodeBuffer, and Asn1XerOutputStream.

1.33 Asn1XerEncoder_Fields Struct Reference

Static Public Attributes

- static readonly int XERDATA = 2
- static readonly int XEREND = 3
- static readonly int XERINDENT = 3
- static readonly int XERINIT = 0
- static readonly int XERSTART = 1

1.33.1 Detailed Description

This class defines the constant variables for Asn1XerEncoder.

1.33.2 Member Data Documentation

1.33.2.1 readonly int XERDATA = 2 [static]

XER characters (data) state

1.33.2.2 readonly int XEREND = 3 [static]

XER end element state

1.33.2.3 readonly int XERINDENT = 3 [static]

Number of indent spaces required to print XER element

1.33.2.4 readonly int XERINIT = 0 [static]

XER initialization state

1.33.2.5 readonly int XERSTART = 1 [static]

XER start element state

1.34 Asn1XerOpenType Class Reference

Public Member Functions

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

1.34.1 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

1.34.2 Constructor & Destructor Documentation

1.34.2.1 Asn1XerOpenType ()

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

1.34.2.2 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.

Parameters

data Byte array containing a previously encoded value.

1.34.2.3 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.

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

1.35 Asn1XerOutputStream Class Reference

Inherits Com::Objsys::Asn1::Runtime::Asn1XerEncoder.

Public Member Functions

- Asn1XerOutputStream (System.IO.Stream os, bool canonical, int bufSize)
- Asn1XerOutputStream (System.IO.Stream os)
- virtual void Copy (System.String data)
- virtual void Copy (byte[ ] data, int off, int len)
- virtual void Copy (byte[ ] data)
- virtual void Copy (byte data)
- virtual void DecrLevel ()
- virtual void EncodeBinStrValue (byte[ ] bits, int nbits)
- virtual void EncodeByte (byte data)
- virtual void EncodeData (System.String data)
- virtual void EncodeEmptyElement (System.String elemName)
- 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 void Write (System.String data)

Properties

- virtual bool Canonical `[set]`
- virtual int State `[get, set]`

1.35.1 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.

1.35.2 Constructor & Destructor Documentation

1.35.2.1 Asn1XerOutputStream (System.IO.Stream *os*)

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

Parameters

os The underlying System.IO.Stream object.

1.35.2.2 Asn1XerOutputStream (System.IO.Stream *os*, bool *canonical*, int *bufSize*)

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

Parameters

os The underlying System.IO.Stream object.

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

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

1.35.3 Member Function Documentation

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

This method copies a character string to the output stream.

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

Parameters

data The string value to copy

Exceptions

Asn1Exception Thrown, if operation is failed.

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

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

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

Parameters

data Array of bytes to copy to the output stream

off The offset in array at which to begin copy.

len The Number of bytes to copy

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.3 virtual void Copy (byte[ ] *data*) `[virtual]`

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

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

Parameters

data Array of bytes to copy to the output stream

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.4 virtual void Copy (byte *data*) [virtual]

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

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

Parameters

data The byte value to copy

1.35.3.5 virtual void DecrLevel () [virtual]

This method decrements the element nesting level counter.

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

This method encodes XML binary string data

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

Parameters

bits Bit String to encode

nbits Number of bits to encode

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.7 virtual void EncodeByte (byte *data*) [virtual]

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

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

Parameters

data The byte value to copy

1.35.3.8 virtual void EncodeData (System.String *data*) `[virtual]`

This method encodes XML string data

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

Parameters

data String value to encode

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.9 virtual void EncodeEmptyElement (System.String *elemName*) `[virtual]`

This method encodes an XML empty element tag.

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

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements Asn1XerEncoder.

1.35.3.10 virtual void EncodeEndDocument () `[virtual]`

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

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

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.11 virtual void EncodeEndElement (System.String *elemName*) `[virtual]`

This method encodes an XML end element tag.

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

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements Asn1XerEncoder.

1.35.3.12 virtual void EncodeHexStrValue (byte[ ] *data*) [virtual]

This method encodes XML hexadecimal string data

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

Parameters

data Data to encode

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.13 virtual void EncodeNamedValue (System.String *valueName*, System.String *elemName*) [virtual]

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

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

Parameters

valueName The named value.

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements Asn1XerEncoder.

1.35.3.14 virtual void EncodeNamedValueElement (System.String *elemName*) [virtual]

This method encodes an XML named value element tag.

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

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.15 virtual void EncodeRealValue (double *data*, System.String *elemName*) [virtual]

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

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

Parameters

data The value to be encoded.

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

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements Asn1XerEncoder.

1.35.3.16 virtual void EncodeStartDocument () [virtual]

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

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

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.17 virtual void EncodeStartElement (System.String *elemName*) [virtual]

This method encodes an XML start element tag.

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

Parameters

elemName The name of element.

Exceptions

Asn1Exception Thrown, if operation is failed.

Implements Asn1XerEncoder.

1.35.3.18 virtual void IncrLevel () [virtual]

This method increments the element nesting level counter.

1.35.3.19 virtual void Indent () [virtual]

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

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

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.3.20 virtual void Write (System.String *data*) `[virtual]`

This method copies a character string to the output stream.

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

Parameters

data The string value to copy

Exceptions

Asn1Exception Thrown, if operation is failed.

1.35.4 Property Documentation

1.35.4.1 virtual bool Canonical `[set]`

Sets the canonical encoding rule.

`Value:` `true` if canonical encoding; otherwise `false`.

1.35.4.2 virtual int State `[get, set]`

Sets the state of the buffer.

`Value:` buffer stat

Implements Asn1XerEncoder.

1.36 Asn1XerSaxHandler Class Reference

Inherits Com::Objsys::Asn1::Runtime::XmlSaxDefaultHandler.

Public Member Functions

- bool ConsumeStartElement (String namespaceURI, String localName, String qName, XmlAttributes atts)
- virtual void EndGroup ()
- override void Error (System.Xml.XmlException exception)
- override void FatalError (System.Xml.XmlException exception)
- virtual void Init (int startLevel)
- bool IsDecodingAsGroup ()
- void SetComplete ()
- override void Warning (System.Xml.XmlException exception)

Protected Member Functions

- internal Asn1XerSaxHandler ()

Protected Attributes

- bool mConsumedStartElement
- internal int mCurrElemID
- internal int mCurrState
- internal int mLevel
- internal readonly int XERDATA = 2
- internal readonly int XEREND = 3
- internal readonly int XERINIT = 0
- internal readonly int XERSTART = 1
- internal readonly int XERUNKNOWN = - 1

Properties

- bool Complete `[get]`
- virtual int State `[get]`

1.36.1 Detailed Description

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

1.36.2 Constructor & Destructor Documentation

1.36.2.1 internal Asn1XerSaxHandler () `[protected]`

The default constructor creates an XER SAX parser instance.

1.36.3 Member Function Documentation

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

This method should be used in preference to invoking StartElement directly when a parent SAX handler is delegating to a child SAX handler.

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.

Exceptions

Asn1XmlSaxUnexpElemExc if certain that some other element is expected.

1.36.3.2 virtual void EndGroup () `[virtual]`

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.

1.36.3.3 override void Error (System.Xml.XmlException *exception*) `[virtual]`

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

Parameters

exception The exception throws by the parser

Exceptions

System.Xml.XmlException The error exception

Reimplemented from XmlSaxDefaultHandler.

1.36.3.4 override void FatalError (System.Xml.XmlException *exception*) `[virtual]`

This method throws a fatal error exception.

Parameters

exception The exception thrown by the parser

Exceptions

System.Xml.XmlException The error exception

Reimplemented from XmlSaxDefaultHandler.

1.36.3.5 virtual void Init (int *startLevel*) `[virtual]`

This method initializes the class member variables.

Parameters

startLevel The XER level to begin

1.36.3.6 bool IsDecodingAsGroup ()

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

1.36.3.7 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).

  Exceptions

  InvalidOperationException if mLevel != mStartLevel.

1.36.3.8 override void Warning (System.Xml.XmlException *exception*) `[virtual]`

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

Parameters

exception The exception Throws by the parser

Exceptions

System.Xml.XmlException The warning exception

Reimplemented from XmlSaxDefaultHandler.

1.36.4 Member Data Documentation

1.36.4.1 bool mConsumedStartElement [protected]

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

1.36.4.2 internal int mCurrElemID [protected]

Variable holds the current element ID

1.36.4.3 internal int mCurrState [protected]

Variable holds the current XER processing state

1.36.4.4 internal int mLevel [protected]

Variable holds the start and current level

1.36.4.5 internal readonly int XERDATA = 2 [protected]

XER characters (data) state

1.36.4.6 internal readonly int XEREND = 3 [protected]

XER end element state

1.36.4.7 internal readonly int XERINIT = 0 [protected]

XER initialization state

1.36.4.8 internal readonly int XERSTART = 1 [protected]

XER start element state

1.36.4.9 internal readonly int XERUNKNOWN = - 1 [protected]

XER unknown state

1.36.5 Property Documentation

1.36.5.1 bool Complete [get]

Returns true when the SAX handler has finished decoding the group.

1.36.5.2 virtual int State `[get]`

Gets the current state of the event handler.

`Value:` current state

1.37 Asn1XerUtil Class Reference

Static Public Member Functions

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

1.37.1 Detailed Description

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

1.37.2 Member Function Documentation

1.37.2.1 static void EncodeReal (Asn1XerEncoder *buffer*, double *value*, System.String *elemName*) [static]

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

Parameters

buffer Encode message buffer object

value Value to be encoded.

elemName Element name

1.38 Asn1XmlEncodeBuffer Class Reference

Public Member Functions

- Asn1XmlEncodeBuffer (int sizeIncrement)
- Asn1XmlEncodeBuffer ()
- 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 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, bool indent)
- virtual void EncodeEndElement (System.String elemName, System.String nsPrefix)
- virtual void EncodeHexStrValue (byte[ ] data)
- virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String nsPrefix)
- virtual void EncodeNamedValueElement (System.String valueName)
- virtual void EncodeStartDocument ()
- virtual void EncodeStartElement (System.String elemName, System.String nsPrefix, bool terminate)
- virtual void EncodeXSIAttrs ()
- virtual void IncrLevel ()
- virtual void Indent ()
- virtual void SetXSIAttrs (Asn1XmlXSIAttrs data)

Properties

- virtual bool Canonical `[set]`
- Asn1XmlEncodeHelper Helper `[get]`
- virtual int State `[get, set]`

1.38.1 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).

1.38.2 Constructor & Destructor Documentation

1.38.2.1 Asn1XmlEncodeBuffer ()

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

1.38.2.2 Asn1XmlEncodeBuffer (int *sizeIncrement*)

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

Parameters

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

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

1.38.3 Member Function Documentation

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

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

Parameters

outs Output will be written to this stream

varName Name of the Decoded ASN1 Type

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

This method copies a character string to the encode buffer.

Parameters

data The string value to copy

1.38.3.3 virtual void DecrLevel () `[virtual]`

This method decrements the element nesting level counter.

1.38.3.4 virtual void EncodeAttr (System.String *qname*, System.String *data*) `[virtual]`

This method encodes an XML attribute value.

Parameters

qname Attribute qualified name.

value Attribute value in string form.

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

This method encodes XML binary string data

Parameters

bits Bit string to encode

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

1.38.3.6 virtual void EncodeByte (byte *data*) [virtual]

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

Parameters

data The byte value to copy

1.38.3.7 virtual void EncodeData (System.String *data*) [virtual]

This method encodes XML string data

Parameters

value String value to encode

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

1.38.3.8 virtual void EncodeDoubleValue (double *data*, System.String *elemName*, System.String *nsPrefix*) [virtual]

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

Parameters

data The value to be encoded.

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

Exceptions

Asn1Exception Thrown, if operation is failed.

1.38.3.9 virtual void EncodeEmptyElement (System.String *elemName*, System.String *nsPrefix*, bool *terminate*) [virtual]

This method encodes an XML empty element tag

Parameters

elemName The name of element.

nsPrefix The namespace prefix of element.

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

1.38.3.10 virtual void EncodeEndDocument () [virtual]

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

1.38.3.11 virtual void EncodeEndElement (System.String *elemName*, System.String *nsPrefix*, bool *indent*) [virtual]

This method encodes an XML end element tag without indenting

Parameters

elemName The name of element.

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

1.38.3.12 virtual void EncodeEndElement (System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an XML end element tag

Parameters

elemName The name of element, as String.

1.38.3.13 virtual void EncodeHexStrValue (byte[ ] *data*) [virtual]

This method encodes XML hexadecimal string data

Parameters

data Data to encode

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

1.38.3.14 virtual void EncodeNamedValue (System.String *valueName*, System.String *elemName*, System.String *nsPrefix*) [virtual]

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

1.38.3.15 virtual void EncodeNamedValueElement (System.String *valueName*) [virtual]

This method encodes an XML named value element tag

Parameters

valueName The named value, as String.

Exceptions

Asn1Exception Thrown, if operation is failed.

1.38.3.16 virtual void EncodeStartDocument () [virtual]

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

1.38.3.17 virtual void EncodeStartElement (System.String *elemName*, System.String *nsPrefix*, bool *terminate*) [virtual]

This method encodes an XML start element tag.

Parameters

elemName The name of element.

nsPrefix The namespace prefix of element.

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

1.38.3.18 virtual void EncodeXSIAttrs () [virtual]

This method encodes XSI attributes.

1.38.3.19 virtual void IncrLevel () [virtual]

This method increments the element nesting level counter.

1.38.3.20 virtual void Indent () [virtual]

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

1.38.3.21 virtual void SetXSIAttrs (Asn1XmlXSIAttrs *data*) [virtual]

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

Parameters

value XSI attributes object

1.38.4 Property Documentation

1.38.4.1 virtual bool Canonical [set]

Sets the canonical encoding rule.

Value: `true` if canonical encoding; otherwise `false`.

1.38.4.2 Asn1XmlEncodeHelper Helper [get]

Gets the encoding helper object.

1.38.4.3 virtual int State `[get, set]`

Sets the state of the buffer.

`Value:` buffer stat

1.39 Asn1XmlOutputStream Class Reference

Public Member Functions

- Asn1XmlOutputStream (System.IO.Stream os, bool canonical, int bufSize)
- Asn1XmlOutputStream (System.IO.Stream os, int bufSize)
- Asn1XmlOutputStream (System.IO.Stream os)
- virtual void Copy (System.String data)
- virtual void Copy (byte[ ] data, int off, int len)
- virtual void Copy (byte[ ] data)
- virtual void Copy (byte data)
- virtual void DecrLevel ()
- virtual void 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, bool indent)
- virtual void EncodeEndElement (System.String elemName, System.String nsPrefix)
- virtual void EncodeHexStrValue (byte[ ] data)
- virtual void EncodeNamedValue (System.String valueName, System.String elemName, System.String nsPrefix)
- virtual void EncodeNamedValueElement (System.String valueName)
- virtual void EncodeStartDocument ()
- virtual void EncodeStartElement (System.String elemName, System.String nsPrefix, bool terminate)
- virtual void EncodeXSIAttrs ()
- virtual void IncrLevel ()
- virtual void Indent ()
- virtual void SetXSIAttrs (Asn1XmlXSIAttrs data)
- virtual void Write (System.String data)

Properties

- virtual bool Canonical `[set]`
- virtual Asn1XmlEncodeHelper Helper `[get]`
- virtual int State `[get, set]`

1.39.1 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.

1.39.2 Constructor & Destructor Documentation

1.39.2.1 Asn1XmlOutputStream (System.IO.Stream *os*)

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

Parameters

os The underlying System.IO.Stream object.

1.39.2.2 Asn1XmlOutputStream (System.IO.Stream *os*, int *bufSize*)

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

Parameters

os The underlying System.IO.Stream object.

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

1.39.2.3 Asn1XmlOutputStream (System.IO.Stream *os*, bool *canonical*, int *bufSize*)

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

Parameters

os The underlying System.IO.Stream object.

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

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

1.39.3 Member Function Documentation

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

This method copies a character string to the output stream.

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>

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

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

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>

1.39.3.3 virtual void Copy (byte[ ] *data*) [virtual]

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

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>

1.39.3.4 virtual void Copy (byte *data*) [virtual]

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

Parameters

value The byte value to copy

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

1.39.3.5 virtual void DecrLevel () [virtual]

This method decrements the element nesting level counter.

1.39.3.6 virtual void EncodeAttr (System.String *qname*, System.String *data*) [virtual]

This method encodes an XML attribute value.

Parameters

qname Attribute qualified name.

value Attribute value in string form.

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

This method encodes XML binary string data

Parameters

bits Bit string to encode

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

1.39.3.8 virtual void EncodeByte (byte *data*) [virtual]

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

Parameters

value The byte value to copy

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

1.39.3.9 virtual void EncodeData (System.String *data*) [virtual]

This method encodes XML string data

Parameters

value String value to encode

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

1.39.3.10 virtual void EncodeDoubleValue (double *data*, System.String *elemName*, System.String *nsPrefix*) [virtual]

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

Parameters

data The value to be encoded.

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

Exceptions

Asn1Exception Thrown, if operation is failed.

1.39.3.11 virtual void EncodeEmptyElement (System.String *elemName*, System.String *nsPrefix*, bool *terminate*) [virtual]

This method encodes an XML empty element tag

Parameters

elemName The name of element.

nsPrefix The namespace prefix of element.

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

1.39.3.12 virtual void EncodeEndDocument () [virtual]

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

1.39.3.13 virtual void EncodeEndElement (System.String *elemName*, System.String *nsPrefix*, bool *indent*) [virtual]

This method encodes an XML end element tag without indenting

Parameters

elemName The name of element.

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

1.39.3.14 virtual void EncodeEndElement (System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes an XML end element tag

Parameters

elemName The name of element, as String.

1.39.3.15 virtual void EncodeHexStrValue (byte[ ] *data*) [virtual]

This method encodes XML hexadecimal string data

Parameters

data Data to encode

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

1.39.3.16 virtual void EncodeNamedValue (System.String *valueName*, System.String *elemName*, System.String *nsPrefix*) [virtual]

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

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>

1.39.3.17 virtual void EncodeNamedValueElement (System.String *valueName*) [virtual]

This method encodes an XML named value element tag

Parameters

valueName The named value, as String.

Exceptions

Asn1Exception Thrown, if operation is failed.

1.39.3.18 virtual void EncodeStartDocument () [virtual]

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

1.39.3.19 virtual void EncodeStartElement (System.String *elemName*, System.String *nsPrefix*, bool *terminate*) [virtual]

This method encodes an XML start element tag.

Parameters

elemName The name of element.

nsPrefix The namespace prefix of element.

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

1.39.3.20 virtual void EncodeXSIAttrs () [virtual]

This method encodes XSI attributes.

1.39.3.21 virtual void IncrLevel () [virtual]

This method increments the element nesting level counter.

1.39.3.22 virtual void Indent () [virtual]

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

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

1.39.3.23 virtual void SetXSIAttrs (Asn1XmlXSIAttrs *data*) [virtual]

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

Parameters

value XSI attributes object

1.39.3.24 virtual void Write (System.String *data*) `[virtual]`

This method copies a character string to the output stream.

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>

1.39.4 Property Documentation

1.39.4.1 virtual bool Canonical `[set]`

Sets the canonical encoding rule.

Value: `true` if canonical encoding; otherwise `false`.

1.39.4.2 virtual Asn1XmlEncodeHelper Helper `[get]`

Gets the encoding helper object.

Value: The helper object.

1.39.4.3 virtual int State `[get, set]`

Sets the state of the buffer.

Value: buffer stat

1.40 Asn1XmlUtil Class Reference

Static Public Member Functions

- static string CaptureElement (System.Xml.XmlReader reader, bool contentOnly, bool injectNsDecls)
- static void EncodeDouble (Asn1XmlEncoder buffer, double data)
- static void EncodeDouble (Asn1XmlEncoder buffer, double data, System.String elemName, System.String nsPrefix)
- static void EncodeNSAttrs (Asn1XmlEncoder buffer, Asn1XmlNamespace[ ] nsArray)
- static double GetMinusZero ()
- static string GetTextContent (System.Xml.XmlReader reader)
- static System.String GetXMLString (System.String data)
- static bool IsMinusZero (double value)
- static void KeepNullsInString (bool keep)
- static String[ ] TokenizeXsdList (String listValue)

1.40.1 Detailed Description

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

1.40.2 Member Function Documentation

1.40.2.1 static string CaptureElement (System.Xml.XmlReader *reader*, bool *contentOnly*, bool *injectNsDecls*) `[static]`

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.

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.

1.40.2.2 static void EncodeDouble (Asn1XmlEncoder *buffer*, double *data*) `[static]`

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

Parameters

buffer Encode message buffer object

value Value to be encoded.

1.40.2.3 static void EncodeDouble (Asn1XmlEncoder *buffer*, double *data*, System.String *elemName*, System.String *nsPrefix*) [static]

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

Parameters

buffer Encode message buffer object

value Value to be encoded.

elemName Element name

nsPrefix Element namespace prefix value

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

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

Parameters

nsArray Array of XML namespace prefix/URI mappings.

1.40.2.5 static double GetMinusZero () [static]

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

Returns

double value for "-0".

1.40.2.6 static string GetTextContent (System.Xml.XmlReader *reader*) [static]

Return the current text node and subsequent text content until an EndElement is found.

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.

1.40.2.7 static System.String GetXMLString (System.String *data*) [static]

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.

Parameters

data String to convert

Returns

Converted string value

1.40.2.8 static bool IsMinusZero (double *value*) `[static]`

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

Parameters

value value to test

Returns

true, if this value is "-0".

1.40.2.9 static void KeepNullsInString (bool *keep*) `[static]`

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.

Parameters

keep Boolean switch to keep or ignore null bytes.

1.40.2.10 static String [ ] TokenizeXsdList (String *listValue*) `[static]`

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

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.

1.41 XmlAttribute Class Reference

Public Member Functions

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

Public Attributes

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

1.41.1 Detailed Description

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

1.41.2 Constructor & Destructor Documentation

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

This is the constructor of the XmlAttribute

Parameters

Uri The namespace URI of the attribute

Lname The local name of the attribute

Qname The long(Qualify) name of attribute

Type The type of the attribute

Value The value of the attribute

1.41.3 Member Data Documentation

1.41.3.1 System.String att_fullName

Variable holds attribte full name (namespace + local name)

1.41.3.2 System.String att_localName

Variable holds attribte local name

1.41.3.3 System.String att_type

Variable holds attribte type

1.41.3.4 System.String att_URI

Variable holds attribte namespace

1.41.3.5 System.String att_value

Variable holds attribte value

1.42 XmlAttributes Class Reference

Classes

- class XmlAttribute

Public Member Functions

- virtual void Add (System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)
- virtual void Clear ()
- virtual System.String GetFullName (int index)
- virtual int GetIndex (System.String Uri, System.String Lname)
- virtual int GetIndex (System.String Qname)
- virtual int GetLength ()
- virtual System.String GetLocalName (int index)
- virtual System.String GetQName (int index)
- virtual System.String GetType (System.String Uri, System.String Lname)
- virtual System.String GetType (System.String Qname)
- virtual System.String GetType (int index)
- virtual System.String GetURI (int index)
- virtual System.String GetValue (System.String Uri, System.String Lname)
- virtual System.String GetValue (System.String Qname)
- virtual System.String GetValue (int index)
- virtual void RemoveAttribute (System.String indexName)
- virtual void RemoveAttribute (int index)
- virtual void SetAttribute (int index, System.String Uri, System.String Lname, System.String Qname, System.String Type, System.String Value)
- virtual void SetAttributes (XmlAttributes Source)
- virtual void SetFullName (int index, System.String FullName)
- virtual void SetLocalName (int index, System.String LocalName)
- virtual void SetType (int index, System.String Type)
- virtual void SetURI (int index, System.String URI)
- virtual void SetValue (int index, System.String Value)
- XmlAttributes (XmlAttributes arrayList)
- XmlAttributes ()

1.42.1 Detailed Description

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

1.42.2 Constructor & Destructor Documentation

1.42.2.1 XmlAttributes ()

Builds a new instance of XmlAttributes.

1.42.2.2 XmlAttributes (XmlAttributes *arrayList*)

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

Parameters

arrayList An ArraList of XmlAttribute class instances.

Returns

A new instance of XmlAttributes

1.42.3 Member Function Documentation

1.42.3.1 virtual void Add (System.String *Uri*, System.String *Lname*, System.String *Qname*, System.String *Type*, System.String *Value*) `[virtual]`

Adds a new attribute elment to the given XmlAttributes instance.

Parameters

Uri The Uri of the attribute to be added.
Lname The Local name of the attribute to be added.
Qname The Long(qualify) name of the attribute to be added.
Type The type of the attribute to be added.
Value The value of the attribute to be added.

1.42.3.2 virtual void Clear () `[virtual]`

Clears the list of attributes in the given AttributesSupport instance.

1.42.3.3 virtual System.String GetFullName (int *index*) `[virtual]`

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

Parameters

index The attribute index.

Returns

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

1.42.3.4 virtual int GetIndex (System.String *Uri*, System.String *Lname*) `[virtual]`

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

Parameters

Uri The namespace URI of the attribute to search.
Lname The local name of the attribute to search.

Returns

An zero-based index of the attribute if it is found, otherwise it returns -1.

1.42.3.5 virtual int GetIndex (System.String *Qname*) [virtual]

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

Parameters

Qname The qualified name of the attribute to search.

Returns

An zero-based index of the attribute if it is found, otherwise it returns -1.

1.42.3.6 virtual int GetLength () [virtual]

Returns the number of attributes saved in the XmlAttributes instance.

Returns

The number of elements in the given XmlAttributes instance.

1.42.3.7 virtual System.String GetLocalName (int *index*) [virtual]

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

Parameters

index The attribute index.

Returns

The local name of the attribute indicated by the index or null if the index is out of bounds.

1.42.3.8 virtual System.String GetQName (int *index*) [virtual]

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

Parameters

index

Returns

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

1.42.3.9 virtual System.String GetType (System.String *Uri*, System.String *Lname*) [virtual]

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

Parameters

Uri The namespace URI of the attribute to search.
Lname The local name of the attribute to search.

Returns

The type of the attribute if it exist otherwise returns null.

1.42.3.10 virtual System.String GetType (System.String *Qname*) `[virtual]`

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

Parameters

Qname The qualified name of the attribute to search.

Returns

The type of the attribute if it exist otherwise returns null.

1.42.3.11 virtual System.String GetType (int *index*) `[virtual]`

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

Parameters

index The attribute index.

Returns

The type of the attribute indicated by the index or null if the index is out of bounds.

1.42.3.12 virtual System.String GetURI (int *index*) `[virtual]`

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

Parameters

index The attribute index.

Returns

The namespace URI of the attribute indicated by the index or null if the index is out of bounds.

1.42.3.13 virtual System.String GetValue (System.String *Uri*, System.String *Lname*) `[virtual]`

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

Parameters

Uri The namespace URI of the attribute to search.

Lname The local name of the attribute to search.

Returns

The value of the attribute if it exist otherwise returns null.

1.42.3.14 virtual System.String GetValue (System.String *Qname*) [virtual]

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

Parameters

Qname The qualified name of the attribute to search.

Returns

The value of the attribute if it exist otherwise returns null.

1.42.3.15 virtual System.String GetValue (int *index*) [virtual]

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

Parameters

index The attribute index.

Returns

The value of the attribute indicated by the index or null if the index is out of bounds.

1.42.3.16 virtual void RemoveAttribute (System.String *indexName*) [virtual]

This method eliminates the XmlAttribute instance in the specified index.

Parameters

indexName The index name of the attribute.

1.42.3.17 virtual void RemoveAttribute (int *index*) [virtual]

This method eliminates the XmlAttribute instance at the specified index.

Parameters

index The index of the attribute.

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

Replaces an XmlAttribute in the given XmlAttributes instance.

Parameters

index The index of the attribute.

Uri The namespace URI of the new XmlAttribute.

Lname The local name of the new XmlAttribute.

Qname The namespace URI of the new XmlAttribute.

Type The type of the new XmlAttribute.

Value The value of the new XmlAttribute.

1.42.3.19 virtual void SetAttributes (XmlAttributes *Source*) [virtual]

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

Parameters

Source The source XmlAttributes instance.

1.42.3.20 virtual void SetFullName (int *index*, System.String *FullName*) [virtual]

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

Parameters

index The attribute index.

FullName The new qualified name for the attribute.

1.42.3.21 virtual void SetLocalName (int *index*, System.String *LocalName*) [virtual]

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

Parameters

index The attribute index.

LocalName The new Local name for the attribute.

1.42.3.22 virtual void SetType (int *index*, System.String *Type*) [virtual]

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

Parameters

index The attribute index.

Type The new type for the attribute.

1.42.3.23 virtual void SetURI (int *index*, System.String *URI*) [virtual]

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

Parameters

index The attribute index.

URI The new namespace URI for the attribute.

1.42.3.24 virtual void SetValue (int *index*, System.String *Value*) [virtual]

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

Parameters

index The attribute index.

Value The new value for the attribute.

1.43 XmlSaxContentHandler Interface Reference

Inherited by XmlSaxDefaultHandler, and XmlSaxParserAdapter.

Public Member Functions

- void Characters (char[ ] ch, int start, int length)
- void EndDocument ()
- void EndElement (System.String namespaceURI, System.String localName, System.String qName)
- void EndPrefixMapping (System.String prefix)
- void IgnorableWhitespace (char[ ] Ch, int Start, int Length)
- void ProcessingInstruction (System.String target, System.String data)
- void SetDocumentLocator (XmlSaxLocator locator)
- void SkippedEntity (System.String name)
- void StartDocument ()
- void StartElement (System.String namespaceURI, System.String localName, System.String qName, XmlAttributes atts)
- void StartPrefixMapping (System.String prefix, System.String uri)

1.43.1 Detailed Description

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

1.43.2 Member Function Documentation

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

This method manage the notification when Characters elements were found.

Parameters

ch The array with the characters found.

start The index of the first position of the characters found.

length Specify how many characters must be read from the array.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.43.2.2 void EndDocument ()

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

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

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

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

Parameters

namespaceURI The namespace URI of the element.

localName The local name of the element.

qName The long (qualified) name of the element.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.43.2.4 void EndPrefixMapping (System.String *prefix*)

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

Parameters

prefix The prefix that ends.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

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

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

Parameters

Ch The array with the ignorable whitespaces.

Start The index in the array with the ignorable whitespace.

Length The length of the whitespaces.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

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

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

Parameters

target The processing instruction target.

data The processing instruction data.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.43.2.7 void SetDocumentLocator (XmlSaxLocator *locator*)

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

Parameters

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

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.43.2.8 void SkippedEntity (System.String *name*)

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

Parameters

name The name of the skipped entity.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.43.2.9 void StartDocument ()

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

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.43.2.10 void StartElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*, XmlAttributes *atts*)

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

Parameters

namespaceURI The namespace uri of the element tag.

localName The local name of the element.

qName The long (qualified) name of the element.

atts The list of attributes of the element.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

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

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

Parameters

prefix The prefix of the area.

uri The namespace URI of the prefix area.

Implemented in XmlSaxDefaultHandler, and XmlSaxParserAdapter.

1.44 XmlSaxDefaultHandler Class Reference

Inherits Com::Objsys::Asn1::Runtime::XmlSaxContentHandler, Com::Objsys::Asn1::Runtime::XmlSaxErrorHandler, and Com::Objsys::Asn1::Runtime::XmlSaxEntityResolver.

Inherited by Asn1XerSaxHandler.

Public Member Functions

- virtual void Characters (char[ ] ch, int start, int length)
- virtual void EndDocument ()
- virtual void EndElement (System.String ns, System.String localName, System.String qName)
- virtual void EndPrefixMapping (System.String prefix)
- virtual void Error (System.Xml.XmlException exception)
- virtual void FatalError (System.Xml.XmlException exception)
- virtual void IgnorableWhitespace (char[ ] chars, int start, int length)
- virtual void ProcessingInstruction (System.String target, System.String data)
- virtual XmlSource ResolveEntity (System.String publicId, System.String systemId)
- virtual void SetDocumentLocator (XmlSaxLocator locator)
- virtual void SkippedEntity (System.String name)
- virtual void StartDocument ()
- virtual void StartElement (System.String ns, System.String localName, System.String qName, XmlAttributes attributes)
- virtual void StartPrefixMapping (System.String prefix, System.String uri)
- virtual void Warning (System.Xml.XmlException exception)

1.44.1 Detailed Description

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

1.44.2 Member Function Documentation

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

This method manage the notification when Characters element were found.

Parameters

ch The array with the characters founds

start The index of the first position of the characters found

length Specify how many characters must be read from the array

Implements XmlSaxContentHandler.

1.44.2.2 virtual void EndDocument () [virtual]

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

Implements XmlSaxContentHandler.

1.44.2.3 virtual void EndElement (System.String *ns*, System.String *localName*, System.String *qName*) [virtual]

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

Parameters

ns The namespace of the element

localName The local name of the element

qName The long name (qualify name) of the element

Implements XmlSaxContentHandler.

1.44.2.4 virtual void EndPrefixMapping (System.String *prefix*) [virtual]

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

Parameters

prefix The prefix that ends

Implements XmlSaxContentHandler.

1.44.2.5 virtual void Error (System.Xml.XmlException *exception*) [virtual]

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

Parameters

exception The exception thrown by the parser

Implements XmlSaxErrorHandler.

Reimplemented in Asn1XerSaxHandler.

1.44.2.6 virtual void FatalError (System.Xml.XmlException *exception*) [virtual]

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

Parameters

exception The exception Thrown by the parser

Implements XmlSaxErrorHandler.

Reimplemented in Asn1XerSaxHandler.

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

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

Parameters

chars The array with the ignorable whitespaces

start The array offset at the ignorable whitespace

length The length of the whitespaces

Implements XmlSaxContentHandler.

1.44.2.8 virtual void ProcessingInstruction (System.String *target*, System.String *data*) `[virtual]`

This method manage the event when a processing instruction were found

Parameters

target The processing instruction target

data The processing instruction data

Implements XmlSaxContentHandler.

1.44.2.9 virtual XmlSource ResolveEntity (System.String *publicId*, System.String *systemId*) `[virtual]`

Allow the application to resolve external entities.

Parameters

publicId The public identifier of the external entity being referenced, or null if none was supplied.

systemId The system identifier of the external entity being referenced.

Returns

A XmlSourceSupport object describing the new input source, or null to request that the parser open a regular URI connection to the system identifier.

Implements XmlSaxEntityResolver.

1.44.2.10 virtual void SetDocumentLocator (XmlSaxLocator *locator*) `[virtual]`

This method is not supported, is include for compatibility

Parameters

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

Implements XmlSaxContentHandler.

1.44.2.11 virtual void SkippedEntity (System.String *name*) `[virtual]`

This method manage the event when a skipped entity were found

Parameters

name The name of the skipped entity

Implements XmlSaxContentHandler.

1.44.2.12 virtual void StartDocument () [virtual]

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

Implements XmlSaxContentHandler.

1.44.2.13 virtual void StartElement (System.String *ns*, System.String *localName*, System.String *qName*, XmlAttributes *attributes*) [virtual]

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

Parameters

ns The namespace uri of the element tag

localName The local name of the element

qName The Qualify (long) name of the element

attributes The list of attributes of the element

Implements XmlSaxContentHandler.

1.44.2.14 virtual void StartPrefixMapping (System.String *prefix*, System.String *uri*) [virtual]

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

Parameters

prefix The prefix of the area

uri The namespace uri of the prefix area

Implements XmlSaxContentHandler.

1.44.2.15 virtual void Warning (System.Xml.XmlException *exception*) [virtual]

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

Parameters

exception The exception Thrown by the parser

Implements XmlSaxErrorHandler.

Reimplemented in Asn1XerSaxHandler.

1.45 XmlSaxEntityResolver Interface Reference

Inherited by XmlSaxDefaultHandler.

Public Member Functions

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

1.45.1 Detailed Description

Basic interface for resolving entities.

1.45.2 Member Function Documentation

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

Allow the application to resolve external entities.

Parameters

publicId The public identifier of the external entity being referenced, or null if none was supplied.

systemId The system identifier of the external entity being referenced.

Returns

A XmlSourceSupport object describing the new input source, or null to request that the parser open a regular URI connection to the system identifier.

Implemented in XmlSaxDefaultHandler.

1.46 XmlSaxErrorHandler Interface Reference

Inherited by XmlSaxDefaultHandler.

Public Member Functions

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

1.46.1 Detailed Description

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

1.46.2 Member Function Documentation

1.46.2.1 void Error (System.Xml.XmlException *exception*)

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

Parameters

exception The exception thrown by the parser.

Implemented in Asn1XerSaxHandler, and XmlSaxDefaultHandler.

1.46.2.2 void FatalError (System.Xml.XmlException *exception*)

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

Parameters

exception The exception thrown by the parser.

Implemented in Asn1XerSaxHandler, and XmlSaxDefaultHandler.

1.46.2.3 void Warning (System.Xml.XmlException *exception*)

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

Parameters

exception The exception thrown by the parser.

Implemented in Asn1XerSaxHandler, and XmlSaxDefaultHandler.

1.47 XmlSaxLexicalHandler Interface Reference

Public Member Functions

- void Comment (char[ ] ch, int start, int length)
- void EndCDATA ()
- void EndDTD ()
- void EndEntity (System.String name)
- void StartCDATA ()
- void StartDTD (System.String name, System.String publicId, System.String systemId)
- void StartEntity (System.String name)

1.47.1 Detailed Description

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

1.47.2 Member Function Documentation

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

This method manage the notification when Characters elements were found.

Parameters

ch The array with the characters found.

start The index of the first position of the characters found.

length Specify how many characters must be read from the array.

1.47.2.2 void EndCDATA ()

This method manage the notification when the end of a CDATA section were found.

1.47.2.3 void EndDTD ()

This method manage the notification when the end of DTD declarations were found.

1.47.2.4 void EndEntity (System.String *name*)

This method report the end of an entity.

Parameters

name The name of the entity that is ending.

1.47.2.5 void StartCDATA ()

This method manage the notification when the start of a CDATA section were found.

1.47.2.6 void StartDTD (System.String *name*, System.String *publicId*, System.String *systemId*)

This method manage the notification when the start of DTD declarations were found.

Parameters

name The name of the DTD entity.

publicId The public identifier.

systemId The system identifier.

1.47.2.7 void StartEntity (System.String *name*)

This method report the start of an entity.

Parameters

name The name of the entity that is ending.

1.48 XmlSaxLocator Interface Reference

Inherited by XmlSaxLocatorImpl.

Public Member Functions

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

1.48.1 Detailed Description

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

1.48.2 Member Function Documentation

1.48.2.1 int GetColumnNumber ()

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

Returns

The column number where the current document event ends.

Implemented in XmlSaxLocatorImpl.

1.48.2.2 int GetLineNumber ()

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

Returns

The line number where the current document event ends.

Implemented in XmlSaxLocatorImpl.

1.48.2.3 System.String GetPublicId ()

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

Returns

The saved public identifier.

Implemented in XmlSaxLocatorImpl.

1.48.2.4 System.String GetSystemId ()

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

Returns

The saved system identifier.

Implemented in XmlSaxLocatorImpl.

1.49 XmlSaxLocatorImpl Class Reference

Inherits Com::Objsys::Asn1::Runtime::XmlSaxLocator.

Public Member Functions

- virtual int GetColumnNumber ()
- virtual int GetLineNumber ()
- virtual System.String GetPublicId ()
- virtual System.String GetSystemId ()
- virtual void SetColumnNumber (int columnNumber)
- virtual void SetLineNumber (int lineNumber)
- virtual void SetPublicId (System.String publicId)
- virtual void SetSystemId (System.String systemId)
- XmlSaxLocatorImpl (XmlSaxLocator locator)
- XmlSaxLocatorImpl ()

1.49.1 Detailed Description

This class is created for emulates the SAX LocatorImpl behaviors.

1.49.2 Constructor & Destructor Documentation

1.49.2.1 XmlSaxLocatorImpl ()

This method returns a new instance of 'XmlSaxLocatorImpl'.

Returns

A new 'XmlSaxLocatorImpl' instance.

1.49.2.2 XmlSaxLocatorImpl (XmlSaxLocator *locator*)

This method returns a new instance of 'XmlSaxLocatorImpl'. Create a persistent copy of the current state of a locator.

Parameters

locator The current state of a locator.

Returns

A new 'XmlSaxLocatorImpl' instance.

1.49.3 Member Function Documentation

1.49.3.1 virtual int GetColumnNumber () `[virtual]`

Return the saved column number.

Returns

The saved column number.

Implements XmlSaxLocator.

1.49.3.2 virtual int GetLineNumber () [virtual]

Return the saved line number.

Returns

The saved line number.

Implements XmlSaxLocator.

1.49.3.3 virtual System.String GetPublicId () [virtual]

This method is not supported, it is included for compatibility. Return the saved public identifier.

Returns

The saved public identifier.

Implements XmlSaxLocator.

1.49.3.4 virtual System.String GetSystemId () [virtual]

This method is not supported, it is included for compatibility. Return the saved system identifier.

Returns

The saved system identifier.

Implements XmlSaxLocator.

1.49.3.5 virtual void SetColumnNumber (int *columnNumber*) [virtual]

Set the column number for this locator.

Parameters

columnNumber The column number.

1.49.3.6 virtual void SetLineNumber (int *lineNumber*) [virtual]

Set the line number for this locator.

Parameters

lineNumber The line number.

1.49.3.7 virtual void SetPublicId (System.String *publicId*) `[virtual]`

This method is not supported, it is included for compatibility. Set the public identifier for this locator.

Parameters

publicId The new public identifier.

1.49.3.8 virtual void SetSystemId (System.String *systemId*) `[virtual]`

This method is not supported, it is included for compatibility. Set the system identifier for this locator.

Parameters

systemId The new system identifier.

1.50 XmlSaxParser Class Reference

Inherited by XmlSaxParserAdapter.

Public Member Functions

- virtual XmlSaxContentHandler GetContentHandler ()
- virtual XmlSaxEntityResolver GetEntityResolver ()
- virtual XmlSaxErrorHandler GetErrorHandler ()
- virtual void Parse (XmlSource source)
- virtual void Parse (System.IO.Stream stream, System.String URI)
- virtual void Parse (System.IO.Stream stream)
- virtual void Parse (System.String filepath)
- virtual void Parse (System.IO.FileInfo filepath)
- virtual void Parse (XmlSource source, XmlSaxContentHandler handler)
- virtual void Parse (System.IO.Stream stream, XmlSaxContentHandler handler, System.String URI)
- virtual void Parse (System.IO.Stream stream, XmlSaxContentHandler handler)
- virtual void Parse (System.String filepath, XmlSaxContentHandler handler)
- virtual void Parse (System.IO.FileInfo filepath, XmlSaxContentHandler handler)
- virtual void SetContentHandler (XmlSaxContentHandler handler)
- virtual void SetDocumentHandler (XmlSaxContentHandler handler)
- virtual void SetEntityResolver (XmlSaxEntityResolver resolver)
- virtual void SetErrorHandler (XmlSaxErrorHandler handler)
- XmlSaxParser ()

Static Public Member Functions

- static XmlSaxParser CloneInstance (XmlSaxParser instance)
- static XmlSaxParser NewInstance ()

Protected Attributes

- XmlSaxContentHandler callBackHandler
- XmlSaxEntityResolver entityResolver
- XmlSaxErrorHandler errorHandler
- XmlSaxLexicalHandler lexical
- XmlSaxLocatorImpl locator
- bool namespaceAllowed
- System.String parserFileName
- System.Xml.XmlTextReader reader

Properties

- bool NamespaceAllowed `[get, set]`

1.50.1 Detailed Description

Emulates the SAX parsers behaviours.

1.50.2 Constructor & Destructor Documentation

1.50.2.1 XmlSaxParser ()

Public constructor for the class.

1.50.3 Member Function Documentation

1.50.3.1 static XmlSaxParser CloneInstance (XmlSaxParser *instance*) [static]

Create a clone instance of 'XmlSaxParser'.

Parameters

instance The `XmlSaxParser` instance to be cloned.

Returns

A clone 'XmlSaxParser' instance.

1.50.3.2 virtual XmlSaxContentHandler GetContentHandler () [virtual]

Obtains the object that will handle all the content events.

Returns

The object that handles the content events.

1.50.3.3 virtual XmlSaxEntityResolver GetEntityResolver () [virtual]

Returns the current entity resolver.

Returns

The current entity resolver, or null if none has been registered.

1.50.3.4 virtual XmlSaxErrorHandler GetErrorHandler () [virtual]

Assigns the object that will handle all the error events.

Returns

The object that handles the error events.

1.50.3.5 static XmlSaxParser NewInstance () [static]

Returns a new instance of 'XmlSaxParser'.

Returns

A new 'XmlSaxParser' instance.

1.50.3.6 virtual void Parse (XmlSource *source*) [virtual]

Parses the specified 'XmlSource' and processes the events over the specified handler, and resolves the entities with the specified URI.

Parameters

source The 'XmlSource' instance with the XML.

1.50.3.7 virtual void Parse (System.IO.Stream *stream*, System.String *URI*) [virtual]

Parses the specified stream and processes the events over previously specified handler, and resolves the external entities with the specified URI.

Parameters

stream The stream with the XML.

URI The namespace URI for resolve external etities.

1.50.3.8 virtual void Parse (System.IO.Stream *stream*) [virtual]

Parses the specified stream and process the events over previusly specified handler.

Parameters

stream The stream with the XML.

1.50.3.9 virtual void Parse (System.String *filepath*) [virtual]

Parses the specified file path and processes the events over previously specified handler.

Parameters

filepath The path of the file with the XML.

1.50.3.10 virtual void Parse (System.IO.FileInfo *filepath*) [virtual]

Parses the specified file and process the events over previously specified handler.

Parameters

filepath The file with the XML.

1.50.3.11 virtual void Parse (XmlSource *source*, XmlSaxContentHandler *handler*) [virtual]

Parses the specified 'XmlSource' instance and process the events over the specified handler, and resolves the entities with the specified URI.

Parameters

source The 'XmlSource' that contains the XML.

handler The handler that manages the parser events.

1.50.3.12 virtual void Parse (System.IO.Stream *stream*, XmlSaxContentHandler *handler*, System.String *URI*) `[virtual]`

Parses the specified stream and process the events over the specified handler, and resolves the entities with the specified URI.

Parameters

stream The stream with the XML.

handler The handler that manage the parser events.

URI The namespace URI for resolve external etities.

1.50.3.13 virtual void Parse (System.IO.Stream *stream*, XmlSaxContentHandler *handler*) `[virtual]`

Parses the specified stream and process the events over the specified handler.

Parameters

stream The stream with the XML.

handler The handler that manage the parser events.

1.50.3.14 virtual void Parse (System.String *filepath*, XmlSaxContentHandler *handler*) `[virtual]`

Parses the specified file path and process the events over the specified handler.

Parameters

filepath The path of the file to be used.

handler The handler that manage the parser events.

1.50.3.15 virtual void Parse (System.IO.FileInfo *filepath*, XmlSaxContentHandler *handler*) `[virtual]`

Parses the specified file and process the events over the specified handler.

Parameters

filepath The file to be used.

handler The handler that manages the parser events.

1.50.3.16 virtual void SetContentHandler (XmlSaxContentHandler *handler*) `[virtual]`

Assigns the object that will handle all the content events.

Parameters

handler The object that handles the content events.

1.50.3.17 virtual void SetDocumentHandler (XmlSaxContentHandler *handler*) [virtual]

Assigns the object that will handle all the document events.

Parameters

handler The object that handles the content events.

1.50.3.18 virtual void SetEntityResolver (XmlSaxEntityResolver *resolver*) [virtual]

Allows an application to register an entity resolver.

Parameters

resolver The entity resolver.

1.50.3.19 virtual void SetErrorHandler (XmlSaxErrorHandler *handler*) [virtual]

Assigns the object that will handle all the error events.

Parameters

handler The object that handles the errors events.

1.50.4 Member Data Documentation

1.50.4.1 XmlSaxContentHandler callBackHandler [protected]

XmlSaxContentHandler variable manages the Content events of a XML document.

1.50.4.2 XmlSaxEntityResolver entityResolver [protected]

XmlSaxEntityResolver variable for resolving entities

1.50.4.3 XmlSaxErrorHandler errorHandler [protected]

XmlSaxErrorHandler variable manages errors during the parsing of a XML document

1.50.4.4 XmlSaxLexicalHandler lexical [protected]

XmlSaxLexicalHandler variable manages the Content events of a XML document

1.50.4.5 XmlSaxLocatorImpl locator [protected]

XmlSaxLocatorImpl variable to emulates the SAX LocatorImpl behaviors.

1.50.4.6 bool namespaceAllowed [protected]

Bool variable manages XML document namespace processing.

1.50.4.7 System.String parserFileName [protected]

String variable holds the XER or XML message file name

1.50.4.8 System.Xml.XmlTextReader reader [protected]

XmlTextReader variable manages XML document parsing.

1.50.5 Property Documentation

1.50.5.1 bool NamespaceAllowed [get, set]

Indicates whether the 'XmlSaxParser' allows namespaces.

1.51 XmlSaxParserAdapter Class Reference

Inherits Com::Objsys::Asn1::Runtime::XmlSaxParser, and Com::Objsys::Asn1::Runtime::XmlSaxContentHandler.

Public Member Functions

- virtual void Characters (char[ ] ch, int start, int length)
- virtual void EndDocument ()
- virtual void EndElement (System.String namespaceURI, System.String localName, System.String qName)
- virtual void EndPrefixMapping (System.String prefix)
- virtual void IgnorableWhitespace (char[ ] ch, int start, int length)
- virtual void ProcessingInstruction (System.String target, System.String data)
- virtual void SetDocumentLocator (XmlSaxLocator locator)
- virtual void SkippedEntity (System.String name)
- virtual void StartDocument ()
- virtual void StartElement (System.String namespaceURI, System.String localName, System.String qName, XmlAttributes qAtts)
- virtual void StartPrefixMapping (System.String prefix, System.String uri)

1.51.1 Detailed Description

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

1.51.2 Member Function Documentation

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

This method manage the notification when Characters element were found.

Parameters

ch The array with the characters founds

start The index of the first position of the characters found

length Specify how many characters must be read from the array

Implements XmlSaxContentHandler.

1.51.2.2 virtual void EndDocument () [virtual]

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

Implements XmlSaxContentHandler.

1.51.2.3 virtual void EndElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*) [virtual]

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

Parameters

namespaceURI The namespace URI of the element

localName The local name of the element

qName The long name (qualify name) of the element

Implements XmlSaxContentHandler.

1.51.2.4 virtual void EndPrefixMapping (System.String *prefix*) `[virtual]`

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

Parameters

prefix The prefix that ends.

Implements XmlSaxContentHandler.

1.51.2.5 virtual void IgnorableWhitespace (char[ ] *ch*, int *start*, int *length*) `[virtual]`

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

Parameters

ch The array with the ignorable whitespaces

start The index in the array with the ignorable whitespace

length The length of the whitespaces

Implements XmlSaxContentHandler.

1.51.2.6 virtual void ProcessingInstruction (System.String *target*, System.String *data*) `[virtual]`

This method manage the event when a processing instruction were found

Parameters

target The processing instruction target

data The processing instruction data

Implements XmlSaxContentHandler.

1.51.2.7 virtual void SetDocumentLocator (XmlSaxLocator *locator*) `[virtual]`

Receive an object for locating the origin of events into the XML document

Parameters

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

Implements XmlSaxContentHandler.

1.51.2.8 virtual void SkippedEntity (System.String *name*) [virtual]

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

Parameters

name The name of the skipped entity.

Implements XmlSaxContentHandler.

1.51.2.9 virtual void StartDocument () [virtual]

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

Implements XmlSaxContentHandler.

1.51.2.10 virtual void StartElement (System.String *namespaceURI*, System.String *localName*, System.String *qName*, XmlAttributes *qAtts*) [virtual]

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

Parameters

namespaceURI The namespace uri of the element tag

localName The local name of the element

qName The Qualify (long) name of the element

qAtts The list of attributes of the element

Implements XmlSaxContentHandler.

1.51.2.11 virtual void StartPrefixMapping (System.String *prefix*, System.String *uri*) [virtual]

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

Parameters

prefix The prefix of the area.

uri The namespace URI of the prefix area.

Implements XmlSaxContentHandler.

1.52 XmlSource Class Reference

Public Member Functions

- XmlSource (System.String source)
- XmlSource (System.IO.StreamReader reader)
- XmlSource (System.IO.Stream stream)
- XmlSource ()

Properties

- System.IO.Stream Bytes `[get, set]`
- System.IO.StreamReader Characters `[get, set]`
- System.String Uri `[get, set]`

1.52.1 Detailed Description

This class is used to encapsulate a source of Xml code in an single class.

1.52.2 Constructor & Destructor Documentation

1.52.2.1 XmlSource ()

Constructs an empty XmlSource instance.

1.52.2.2 XmlSource (System.IO.Stream *stream*)

Constructs a XmlSource instance with the specified source System.IO.Stream.

Parameters

stream The stream containing the document.

1.52.2.3 XmlSource (System.IO.StreamReader *reader*)

Constructs a XmlSource instance with the specified source System.IO.StreamReader.

Parameters

reader The reader containing the document.

1.52.2.4 XmlSource (System.String *source*)

Construct a XmlSource instance with the specified source Uri string.

Parameters

source The source containing the document.

1.52.3 Property Documentation

1.52.3.1 System.IO.Stream Bytes `[get, set]`

Represents the source Stream of the XmlSource.

1.52.3.2 System.IO.StreamReader Characters `[get, set]`

Represents the source StreamReader of the XmlSource.

1.52.3.3 System.String Uri `[get, set]`

Represents the source URI of the XmlSource.

Index