

# ASN1C

---

ASN.1 Compiler  
Version 6.1  
C# Common Runtime Classes  
Reference Manual



The software described in this document is furnished under a license agreement and may be used only in accordance with the terms of this agreement.

### **Copyright Notice**

Copyright ©1997-2008 Objective Systems, Inc. All rights reserved.

This document may be distributed in any form, electronic or otherwise, provided that it is distributed in its entirety and that the copyright and this notice are included.

### **Author's Contact Information**

Comments, suggestions, and inquiries regarding ASN1C may be submitted via electronic mail to [info@obj-sys.com](mailto:info@obj-sys.com).



# Contents

<b>1</b>	<b>ASN1C C# Runtime Library</b>	<b>1</b>
<b>2</b>	<b>ASN1C C# Runtime Library Part I Namespace Documentation</b>	<b>3</b>
2.1	Package Com.Objsys.Asn1.Runtime . . . . .	3
<b>3</b>	<b>ASN1C C# Runtime Library Part I Class Documentation</b>	<b>5</b>
3.1	Asn18BitCharString Class Reference . . . . .	5
3.2	Asn1BigInteger Class Reference . . . . .	10
3.3	Asn1BitString Class Reference . . . . .	16
3.4	Asn1BMPString Class Reference . . . . .	26
3.5	Asn1Boolean Class Reference . . . . .	31
3.6	Asn1CharRange Class Reference . . . . .	37
3.7	Asn1CharSet Class Reference . . . . .	39
3.8	Asn1CharString Class Reference . . . . .	42
3.9	Asn1Choice Class Reference . . . . .	48
3.10	Asn1ChoiceExt Class Reference . . . . .	51
3.11	Asn1ConsVioException Class Reference . . . . .	53
3.12	Asn1DecodeBuffer Class Reference . . . . .	54
3.13	Asn1DiscreteCharSet Class Reference . . . . .	60
3.14	Asn1EncodeBuffer Class Reference . . . . .	62
3.15	Asn1EndOfBufferException Class Reference . . . . .	66
3.16	Asn1Enumerated Class Reference . . . . .	67
3.17	Asn1Exception Class Reference . . . . .	73
3.18	Asn1GeneralizedTime Class Reference . . . . .	74
3.19	Asn1GeneralString Class Reference . . . . .	79
3.20	Asn1GraphicString Class Reference . . . . .	81
3.21	Asn1IA5String Class Reference . . . . .	83
3.22	Asn1InputStream Interface Reference . . . . .	85
3.23	Asn1Integer Class Reference . . . . .	87

3.24	<a href="#">Asn1InvalidArgException Class Reference</a>	97
3.25	<a href="#">Asn1InvalidChoiceOptionException Class Reference</a>	98
3.26	<a href="#">Asn1InvalidEnumException Class Reference</a>	99
3.27	<a href="#">Asn1InvalidLengthException Class Reference</a>	100
3.28	<a href="#">Asn1InvalidObjectIDException Class Reference</a>	101
3.29	<a href="#">Asn1MessageBuffer Class Reference</a>	102
3.30	<a href="#">Asn1MissingRequiredException Class Reference</a>	104
3.31	<a href="#">Asn1NamedEventHandler Interface Reference</a>	105
3.32	<a href="#">Asn1Null Class Reference</a>	107
3.33	<a href="#">Asn1NumericString Class Reference</a>	111
3.34	<a href="#">Asn1ObjectDescriptor Class Reference</a>	115
3.35	<a href="#">Asn1ObjectIdentifier Class Reference</a>	117
3.36	<a href="#">Asn1OctetString Class Reference</a>	122
3.37	<a href="#">Asn1OpenExt Class Reference</a>	130
3.38	<a href="#">Asn1OpenType Class Reference</a>	135
3.39	<a href="#">Asn1OutputStream Class Reference</a>	139
3.40	<a href="#">Asn1PrintableString Class Reference</a>	144
3.41	<a href="#">Asn1Real Class Reference</a>	146
3.42	<a href="#">Asn1RelativeOID Class Reference</a>	151
3.43	<a href="#">Asn1SeqOrderException Class Reference</a>	155
3.44	<a href="#">Asn1Status Class Reference</a>	156
3.45	<a href="#">Asn1T61String Class Reference</a>	157
3.46	<a href="#">Asn1Tag Class Reference</a>	159
3.47	<a href="#">Asn1Time Class Reference</a>	164
3.48	<a href="#">Asn1TraceHandler Class Reference</a>	179
3.49	<a href="#">Asn1Type Class Reference</a>	181
3.50	<a href="#">Asn1TypeIF Interface Reference</a>	194
3.51	<a href="#">Asn1UniversalString Class Reference</a>	200
3.52	<a href="#">Asn1UTCTime Class Reference</a>	211
3.53	<a href="#">Asn1UTF8String Class Reference</a>	217
3.54	<a href="#">Asn1Util Class Reference</a>	221
3.55	<a href="#">Asn1Value Class Reference</a>	227
3.56	<a href="#">Asn1ValueParseException Class Reference</a>	228
3.57	<a href="#">Asn1VarWidthCharString Class Reference</a>	229
3.58	<a href="#">Asn1VideotexString Class Reference</a>	232
3.59	<a href="#">Asn1VisibleString Class Reference</a>	234
3.60	<a href="#">BigInteger Class Reference</a>	236

3.61 BooleanHolder Class Reference . . . . .	240
3.62 Diag Class Reference . . . . .	241
3.63 IntHolder Class Reference . . . . .	245
3.64 StringBufferExt Class Reference . . . . .	246
3.65 Tokenizer Class Reference . . . . .	247



# Chapter 1

## ASN1C C# Runtime Library

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

- BER ( As per ITU-T X.690 standard )
- CER ( As per ITU-T X.690 standard )
- DER ( As per ITU-T X.690 standard )
- PER ( As per ITU-T X.691 standard )
- XER ( As per ITU-T X.693 standard )
- XML ( As per asn2xsd converter )



## Chapter 2

# ASN1C C# Runtime Library Part I Namespace Documentation

## 2.1 Package `Com.Objsys.Asn1.Runtime`

### 2.1.1 Detailed Description

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

- BER ( As per ITU-T X.690 standard )
- CER ( As per ITU-T X.690 standard )
- DER ( As per ITU-T X.690 standard )
- PER ( As per ITU-T X.691 standard )
- XER ( As per ITU-T X.693 standard )
- XML ( As per asn2xsd converter )

### Classes

- class [Asn18BitCharString](#)
- class [Asn1BigInteger](#)
- class [Asn1BitString](#)
- class [Asn1BMPString](#)
- class [Asn1Boolean](#)
- class [Asn1CharRange](#)
- class [Asn1CharSet](#)
- class [Asn1CharString](#)
- class [Asn1Choice](#)
- class [Asn1ChoiceExt](#)
- class [Asn1ConsVioException](#)
- class [Asn1DecodeBuffer](#)
- class [Asn1DiscreteCharSet](#)

- class [Asn1EncodeBuffer](#)
- class [Asn1EndOfBufferException](#)
- class [Asn1Enumerated](#)
- class [Asn1Exception](#)
- class [Asn1GeneralizedTime](#)
- class [Asn1GeneralString](#)
- class [Asn1GraphicString](#)
- class [Asn1IA5String](#)
- interface [Asn1InputStream](#)
- class [Asn1Integer](#)
- class [Asn1InvalidArgException](#)
- class [Asn1InvalidChoiceOptionException](#)
- class [Asn1InvalidEnumException](#)
- class [Asn1InvalidLengthException](#)
- class [Asn1InvalidObjectIDException](#)
- class [Asn1MessageBuffer](#)
- class [Asn1MissingRequiredException](#)
- interface [Asn1NamedEventHandler](#)
- class [Asn1Null](#)
- class [Asn1NumericString](#)
- class [Asn1ObjectDescriptor](#)
- class [Asn1ObjectIdentifier](#)
- class [Asn1OctetString](#)
- class [Asn1OpenExt](#)
- class [Asn1OpenType](#)
- class [Asn1OutputStream](#)
- class [Asn1PrintableString](#)
- class [Asn1Real](#)
- class [Asn1RelativeOID](#)
- class [Asn1SeqOrderException](#)
- class [Asn1Status](#)
- class [Asn1T61String](#)
- class [Asn1Tag](#)
- class [Asn1Time](#)
- class [Asn1TraceHandler](#)
- class [Asn1Type](#)
- interface [Asn1TypeIF](#)
- class [Asn1UniversalString](#)
- class [Asn1UTCTime](#)
- class [Asn1UTF8String](#)
- class [Asn1Util](#)
- class [Asn1Value](#)
- class [Asn1ValueParseException](#)
- class [Asn1VarWidthCharString](#)
- class [Asn1VideotexString](#)
- class [Asn1VisibleString](#)
- class [BigInteger](#)
- class [BooleanHolder](#)
- class [Diag](#)
- class [IntHolder](#)
- class [StringBufferExt](#)
- class [Tokenizer](#)

## Chapter 3

# ASN1C C# Runtime Library Part I Class Documentation

### 3.1 Asn18BitCharString Class Reference

Inherits [Asn1CharString](#).

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

#### 3.1.1 Detailed Description

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

#### Public Member Functions

- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerOutputStream outs)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)

#### Public Attributes

- const int [BITSPERCHAR\\_A](#) = 8
- const int [BITSPERCHAR\\_U](#) = 7

#### Protected Member Functions

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

## 3.1.2 Constructor & Destructor Documentation

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

The default constructor creates an empty string object.

#### Parameters:

*typeCode* Universal ID code for ASN.1 character string

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

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

#### Parameters:

*data* Character string

*typeCode* Universal ID code for ASN.1 character string

## 3.1.3 Member Function Documentation

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

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

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

#### Parameters:

*buffer* Decode message buffer object

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

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

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

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

#### Parameters:

*buffer* Decode message buffer object

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

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

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

#### Parameters:

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1NumericString](#), and [Asn1Time](#).

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

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

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

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

#### Parameters:

*outs* PER Encode message stream object

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

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

#### Exceptions:

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

### 3.1.3.5 virtual void Encode (Asn1PerOutputStream *outs*, [Asn1CharSet](#) *charSet*) [virtual]

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

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

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

#### Parameters:

*outs* PER Encode message stream object

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

#### Exceptions:

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

### 3.1.3.6 **override void Encode (Asn1PerOutputStream *outs*)** [virtual]

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

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

Also throws any exception thrown by the `Asn1PerOutputStream`.

#### **Parameters:**

*outs* PER Encode message stream object

#### **Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from `Asn1Type`.

Reimplemented in `Asn1Time`.

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

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

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

#### **Parameters:**

*buffer* Encode message buffer object

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

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

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

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

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

#### **Parameters:**

*buffer* Encode message buffer object

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

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

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

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

**Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1NumericString](#), and [Asn1Time](#).

### **3.1.4 Member Data Documentation**

#### **3.1.4.1 const int [BITSPERCHAR\\_A](#) = 8**

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

#### **3.1.4.2 const int [BITSPERCHAR\\_U](#) = 7**

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

## 3.2 Asn1BigInteger Class Reference

Inherits [Asn1Type](#).

### 3.2.1 Detailed Description

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

### Public Member Functions

- [Asn1BigInteger](#) (System.String value, int radix)
- [Asn1BigInteger](#) (System.String value)
- [Asn1BigInteger](#) ([BigInteger](#) value)
- [Asn1BigInteger](#) ()
- override void [Decode](#) ([Asn1PerDecodeBuffer](#) buffer)
- override void [Decode](#) ([Asn1BerDecodeBuffer](#) buffer, bool explicitTagging, int implicitLength)
- [BigInteger](#) [DecodeValue](#) ([Asn1DecodeBuffer](#) buffer, int length)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) ([Asn1PerOutputStream](#) outs)
- override void [Encode](#) ([Asn1BerOutputStream](#) outs, bool explicitTagging)
- override void [Encode](#) ([Asn1XmlEncoder](#) buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) ([Asn1XerEncoder](#) buffer, System.String elemName)
- override void [Encode](#) ([Asn1PerEncodeBuffer](#) buffer)
- override int [Encode](#) ([Asn1BerEncodeBuffer](#) buffer, bool explicitTagging)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (long value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

### Public Attributes

- [BigInteger](#) [mValue](#)

### Static Public Attributes

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

### Static Protected Member Functions

- static internal int [EncodeValue](#) ([Asn1EncodeBuffer](#) buffer, [BigInteger](#) ivalue, bool doCopy)

## 3.2.2 Constructor & Destructor Documentation

### 3.2.2.1 [Asn1BigInteger](#) ()

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

### 3.2.2.2 [Asn1BigInteger](#) ([BigInteger](#) *value*)

This constructor assigns a big integer object.

#### Parameters:

*value* [BigInteger](#) value

### 3.2.2.3 [Asn1BigInteger](#) ([System.String](#) *value*)

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

#### Parameters:

*value* String value

### 3.2.2.4 [Asn1BigInteger](#) ([System.String](#) *value*, [int](#) *radix*)

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

#### Parameters:

*value* String value

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

## 3.2.3 Member Function Documentation

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

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

#### Parameters:

*buffer* PER Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged  
*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

### 3.2.3.3 **BigInteger DecodeValue** ([Asn1DecodeBuffer](#) *buffer*, int *length*)

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

#### **Parameters:**

*buffer* Decode message buffer object  
*length* Length of encoded contents

#### **Returns:**

Decoded integer value

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

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

#### **Parameters:**

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

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

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

#### **Parameters:**

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

### 3.2.3.6 **override void Encode** ([Asn1PerOutputStream](#) *outs*) [virtual]

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

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

#### **Parameters:**

*outs* PER Output Stream object

#### **Exceptions:**

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

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object  
*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object  
*elemName* Element name  
*nsPrefix* Element namespace prefix name

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object  
*elemName* Element name

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* PER Encode message buffer object

#### Returns:

Length of component or negative status value

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

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

#### **Returns:**

Length of component or negative status value

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

*buffer* Encode message buffer object  
*ivalue* Integer value to encode  
*doCopy* Encode the copy of the value

#### **Returns:**

Length of encoded component

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

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

#### **Parameters:**

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

#### **Returns:**

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

### 3.2.3.14 **virtual bool Equals** ([long](#) *value*) [virtual]

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

#### **Parameters:**

*value* The long value to compare with the current Object.

#### **Returns:**

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

### 3.2.3.15 override int GetHashCode ()

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

#### Returns:

A hash code for the current Object.

### 3.2.3.16 override System.String ToString ()

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

#### Returns:

Stringified representation of the value

## 3.2.4 Member Data Documentation

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

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

Reimplemented from [Asn1Type](#).

### 3.2.4.2 [BigInteger mValue](#)

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

## 3.3 Asn1BitString Class Reference

Inherits [Asn1Type](#).

### 3.3.1 Detailed Description

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

#### Public Types

- enum [StringFormat](#)

#### Public Member Functions

- [Asn1BitString](#) (System.Collections.BitArray bitArray)
- [Asn1BitString](#) (System.String value\_)
- [Asn1BitString](#) (bool[] bitValues)
- [Asn1BitString](#) (int numbits\_, byte[] data)
- [Asn1BitString](#) ()
- virtual void [Clear](#) (int bitno)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix, System.String[] namedbits, int[] namedbitindex)
- virtual void [Encode](#) (Asn1XerEncoder buffer, System.String elemName, System.String[] namedbits, int[] namedbitindex)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (int nbits, byte[] value)
- virtual bool [Get](#) (int bitno)
- override int [GetHashCode](#) ()
- virtual bool [IsNamedBitStr](#) (System.String buffer)
- virtual void [Set](#) (int bitno)
- virtual void [Set](#) (int bitno, bool value)
- virtual bool[] [ToBoolArray](#) ()
- virtual System.String [ToHexString](#) ()
- override System.String [ToString](#) ()

## Public Attributes

- [NonSerialized] byte[] [mValue](#)
- [NonSerialized] int [numbits](#)

## Static Public Attributes

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

## Properties

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

## 3.3.2 Member Enumeration Documentation

### 3.3.2.1 enum [StringFormat](#)

Defines possible string fomrats.

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

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

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

## 3.3.3 Constructor & Destructor Documentation

### 3.3.3.1 [Asn1BitString](#) ()

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

### 3.3.3.2 [Asn1BitString](#) (int *numbits\_*, byte[] *data*)

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

#### Parameters:

- numbits\_* Number of bits
- data* Binary bit string contents

### 3.3.3.3 [Asn1BitString](#) (bool[] *bitValues*)

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

#### Parameters:

- bitValues* The boolean array

#### 3.3.3.4 [Asn1BitString](#) (System.String *value\_*)

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

Examples of valid value formats are as follows:

Binary string: '11010010111001'B

Hex string: '0fa56920014abc'H

##### Parameters:

*value\_* The ASN.1 value specification text

#### 3.3.3.5 [Asn1BitString](#) (System.Collections.BitArray *bitArray*)

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

##### Parameters:

*bitArray* C# BitArray object

### 3.3.4 Member Function Documentation

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

This method clears the given bit in the bit string.

##### Parameters:

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

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

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

##### Parameters:

*buffer* Decode message buffer object

*lower* Lower bound (inclusive) of size constraint

*upper* Upper bound (inclusive) of size constraint

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

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

##### Parameters:

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

##### **Parameters:**

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

Reimplemented from [Asn1Type](#).

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

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

##### **Parameters:**

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

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

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

##### **Parameters:**

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

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

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

##### **Parameters:**

- outs* PER Output Stream object
- lower* Lower bound (inclusive) of size constraint
- upper* Upper bound (inclusive) of size constraint

##### **Exceptions:**

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

#### 3.3.4.8 override void Encode (Asn1PerOutputStream outs) [virtual]

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

##### Parameters:

*outs* PER Output Stream object

##### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

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

##### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

##### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

##### Parameters:

*buffer* Encode message buffer object

*elemName* XML element name used to wrap string

*nsPrefix* XML element namespace name

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object  
*elemName* XML element name used to wrap string  
*nsPrefix* XML element namespace name  
*namedbits* Array of named bits  
*namedbitindex* Array of named bits index values

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

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

**Parameters:**

*buffer* Encode message buffer object  
*elemName* XML element name used to wrap string  
*namedbits* Array of named bits  
*namedbitindex* Array of named bits index values

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

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

**Parameters:**

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

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

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

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

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

**Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

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

#### **Returns:**

Length of component or negative status value

Reimplemented from [Asn1Type](#).

### 3.3.4.17 **override bool Equals (System.Object *value*)**

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

#### **Parameters:**

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

#### **Returns:**

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

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

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

#### **Parameters:**

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

#### **Returns:**

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

### 3.3.4.19 **virtual bool Get (int *bitno*)** [virtual]

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

#### **Parameters:**

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

#### **Returns:**

true if bit is set; otherwise false.

#### 3.3.4.20 **override int GetHashCode ()**

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

##### **Returns:**

A hash code for the current Object.

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

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

##### **Parameters:**

*buffer* Bit string as string to be tested.

##### **Returns:**

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

#### 3.3.4.22 **virtual void Set (int *bitno*) [virtual]**

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

##### **Parameters:**

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

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

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

##### **Parameters:**

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

*value* The Boolean value to assign to the bit.

#### 3.3.4.24 **virtual bool [] ToBoolArray () [virtual]**

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

##### **Returns:**

Boolean array value

#### 3.3.4.25 virtual System.String ToHexString () [virtual]

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

##### Returns:

Stringified representation of the value

#### 3.3.4.26 override System.String ToString ()

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

##### Returns:

String representation of the value

### 3.3.5 Member Data Documentation

#### 3.3.5.1 new readonly Asn1Tag \_TAG [static]

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

Reimplemented from [Asn1Type](#).

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

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

#### 3.3.5.3 [NonSerialized] byte [] mValue

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

#### 3.3.5.4 [NonSerialized] int numbits

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

### 3.3.6 Property Documentation

#### 3.3.6.1 override int Length [get]

Gets the length of the BIT STRING in bits.

Value: Number of bits.

Reimplemented from [Asn1Type](#).

### 3.3.6.2 virtual bool this[int bitno] () [get, set]

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

#### Parameters:

*bitno* The position of the bit in bit array

Value: true if bit is set; otherwise false.

## 3.4 Asn1BMPString Class Reference

Inherits [Asn1CharString](#).

### 3.4.1 Detailed Description

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

#### Public Member Functions

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

#### Public Attributes

- const int [BITSPERCHAR](#) = 16

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 [Asn1BMPString](#) ()

The default constructor creates an empty string object.

#### 3.4.2.2 [Asn1BMPString](#) (System.String data)

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

#### Parameters:

*data* string representation of BMPString

### 3.4.3 Member Function Documentation

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

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

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

##### Parameters:

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

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

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

##### Parameters:

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

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

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

##### Parameters:

- buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

##### Parameters:

- buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

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

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

#### Parameters:

*outs* PER Output Stream object

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

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

#### Exceptions:

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

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

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

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

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

#### Parameters:

*outs* PER Output Stream object

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

#### Exceptions:

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

### 3.4.3.7 override void Encode (Asn1PerOutputStream outs) [virtual]

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

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

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

**Parameters:**

*outs* PER Output Stream object

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

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

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

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

**Parameters:**

*buffer* Encode message buffer object

*charSet* Object representing the permitted alphabet constraint character set

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

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

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

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

**Parameters:**

*buffer* Encode message buffer object

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

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

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

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

##### **Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

##### **Parameters:**

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

##### **Returns:**

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

### 3.4.4 **Member Data Documentation**

#### 3.4.4.1 **new readonly [Asn1Tag\\_TAG](#)** [static]

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

Reimplemented from [Asn1Type](#).

#### 3.4.4.2 **const int [BITSPERCHAR](#) = 16**

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

## 3.5 Asn1Boolean Class Reference

Inherits [Asn1Type](#).

### 3.5.1 Detailed Description

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

#### Public Member Functions

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

#### Static Public Member Functions

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

#### Public Attributes

- [NonSerialized] bool [mValue](#)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)
- static readonly [Asn1Boolean\\_FALSE\\_VALUE](#) = new [Asn1Boolean](#)(false)
- static readonly [Asn1Boolean\\_TRUE\\_VALUE](#) = new [Asn1Boolean](#)(true)

## 3.5.2 Constructor & Destructor Documentation

### 3.5.2.1 [Asn1Boolean](#) ()

The default constructor sets the boolean value to false.

### 3.5.2.2 [Asn1Boolean](#) (bool *value\_*)

This constructor creates a boolean object from a boolean value.

#### Parameters:

*value\_* Boolean value

## 3.5.3 Member Function Documentation

### 3.5.3.1 `override void Decode (Asn1PerDecodeBuffer buffer)` [virtual]

**This method decodes an ASN.1 boolean value using**

the Packed Encoding Rules (PER).

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

in this object.

#### Parameters:

*buffer* PER Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

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

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

in this object.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

#### Returns:

Decoded boolean value

Reimplemented from [Asn1Type](#).

### 3.5.3.3 `virtual void DecodeXER (System.String buffer, System.String attrs)` [virtual]

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

**Parameters:**

*outs* PER Encode message stream object

**Exceptions:**

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

Reimplemented from [Asn1Type](#).

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

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

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

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

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

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object  
*elemName* Element name  
*nsPrefix* Element namespace prefix value

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

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

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

**Parameters:**

*buffer* Encode message buffer object  
*elemName* Element name

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* PER Encode message buffer object

**Returns:**

Length of component or negative status value

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

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

#### **Returns:**

Length (in octets) of encoded component

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

*buffer* Encode message buffer object  
*attrName* Element name

Reimplemented from [Asn1Type](#).

### 3.5.3.13 **override bool Equals (System.Object *value*)**

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

#### **Parameters:**

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

#### **Returns:**

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

### 3.5.3.14 **virtual bool Equals (bool *value*)** [virtual]

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

#### **Parameters:**

*value* The bool value to compare with the current Object.

#### **Returns:**

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

### 3.5.3.15 **override int GetHashCode ()**

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

#### **Returns:**

A hash code for the current Object.

### 3.5.3.16 **static void setTrueEncodedByte (byte *b*) [static]**

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

#### **Parameters:**

*b* The byte value to set.

### 3.5.3.17 **override System.String ToString ()**

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

#### **Returns:**

Stringified representation of the value

## 3.5.4 **Member Data Documentation**

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

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

Reimplemented from [Asn1Type](#).

### 3.5.4.2 **readonly Asn1Boolean FALSE\_VALUE = new Asn1Boolean(false) [static]**

The FALSE\_VALUE constant represents a boolean FALSE value.

### 3.5.4.3 **[NonSerialized] bool mValue**

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

### 3.5.4.4 **readonly Asn1Boolean TRUE\_VALUE = new Asn1Boolean(true) [static]**

The TRUE\_VALUE constant represents a boolean TRUE value.

## 3.6 Asn1CharRange Class Reference

Inherits [Asn1CharSet](#).

### 3.6.1 Detailed Description

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

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

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

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

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

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

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

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

### Public Member Functions

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

### Protected Attributes

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

### Properties

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

## 3.6.2 Constructor & Destructor Documentation

### 3.6.2.1 [Asn1CharRange](#) (int lower, int upper)

This constructor sets the range values.

#### Parameters:

*lower* Range lower value

*upper* Range upper value

### 3.6.3 Member Function Documentation

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

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

**Parameters:**

*index* Index of character within the character set

**Returns:**

Character at given index

**Exceptions:**

[AsnIConsVioException](#) Index not within define range

Implements [Asn1CharSet](#).

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

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

**Parameters:**

*charValue* Character value to search for

**Returns:**

Index of character

**Exceptions:**

[AsnIConsVioException](#) Character not found in set

Implements [Asn1CharSet](#).

### 3.6.4 Member Data Documentation

#### 3.6.4.1 internal int **mLower** [protected]

This variable represents the lower value of the range.

#### 3.6.4.2 internal int **mUpper** [protected]

This variable represents the upper value of the range.

### 3.6.5 Property Documentation

#### 3.6.5.1 override int **MaxValue** [get]

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

Value: Upper Bound Character or Character with max int value

Reimplemented from [Asn1CharSet](#).

## 3.7 Asn1CharSet Class Reference

Inherited by [Asn1CharRange](#), and [Asn1DiscreteCharSet](#).

### 3.7.1 Detailed Description

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

#### Public Member Functions

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

#### Protected Member Functions

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

#### Protected Attributes

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

#### Properties

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

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 internal [Asn1CharSet](#) (int *nchars*) [protected]

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

##### Parameters:

*nchars* Number of characters in the character set

### 3.7.3 Member Function Documentation

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

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

##### Parameters:

*index* Index of character within the character set

**Returns:**

Character at given index

**Exceptions:**

*Asn1ConsVioException* Index not within define range

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

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

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

**Parameters:**

*charValue* Character value to search for

**Returns:**

Index of character

**Exceptions:**

*Asn1ConsVioException* Character not found in set

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

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

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

**Parameters:**

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

**Returns:**

Number of bits-per-character

**3.7.4 Member Data Documentation****3.7.4.1 internal int [mABitsPerChar](#) [protected]**

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

**3.7.4.2 internal int [mUBitsPerChar](#) [protected]**

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

## 3.7.5 Property Documentation

### 3.7.5.1 abstract int MaxValue [get]

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

Value: Upper Bound Character or Character with max int value

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

## 3.8 Asn1CharString Class Reference

Inherits [Asn1Type](#).

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

### 3.8.1 Detailed Description

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

#### Public Member Functions

- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override bool [Equals](#) (System.Object value)
- bool [Equals](#) (System.String value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

#### Public Attributes

- [NonSerialized] System.String [mValue](#)

#### Protected Member Functions

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

#### Protected Attributes

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

#### Properties

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

## 3.8.2 Constructor & Destructor Documentation

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

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

#### Parameters:

*typeCode* Universal ID code for ASN.1 character string

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

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

#### Parameters:

*data* Character string

*typeCode* Universal ID code for ASN.1 character string

## 3.8.3 Member Function Documentation

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

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

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

#### Parameters:

*buffer* Decode message buffer object

*abpc* Number of bits per character (aligned)

*ubpc* Number of bits per character (unaligned)

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

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

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

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

#### Parameters:

*buffer* Decode message buffer object

*abpc* Number of bits per character (aligned)

*ubpc* Number of bits per character (unaligned)

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

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

*tag* Universal tag to apply

Reimplemented in [Asn1Time](#).

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

This method decodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc. using the XML Schema encoding rules(asn2xsd).

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

Reimplemented in [Asn1Time](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*elemName* XML element name used to wrap string

*nsPrefix* XML element namespace value

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1GeneralizedTime](#), and [Asn1UTCTime](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*elemName* XML element name used to wrap string

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1GeneralizedTime](#), and [Asn1UTCTime](#).

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

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

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

#### Parameters:

*buffer* Encode message buffer object

*abpc* Number of bits per character (aligned)

*ubpc* Number of bits per character (unaligned)

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

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

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

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

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

#### Parameters:

*buffer* Encode message buffer object

*abpc* Number of bits per character (aligned)

*ubpc* Number of bits per character (unaligned)

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

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

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

**Parameters:**

*buffer* Encode message buffer object  
*explicitTagging* Flag indicating explicit tagging should be done  
*tag* Universal tag to apply

**Returns:**

Length in octets of encoded component

Reimplemented in [Asn1Time](#).

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

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

**Parameters:**

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

**Returns:**

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

Reimplemented in [Asn1Time](#).

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

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

**Parameters:**

*value* The String value to compare with the current Object.

**Returns:**

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

**3.8.3.13 override int GetHashCode ()**

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

**Returns:**

A hash code for the current Object.

Reimplemented in [Asn1Time](#).

#### 3.8.3.14 **override System.String ToString ()**

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

##### **Returns:**

Stringified representation of the value

### 3.8.4 **Member Data Documentation**

#### 3.8.4.1 **[NonSerialized] internal System.Text.StringBuilder [mStringBuffer](#) [protected]**

The `mStringBuffer` member variable is used to do internal operations on a string being encoded or decoded.

#### 3.8.4.2 **[NonSerialized] System.String [mValue](#)**

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

### 3.8.5 **Property Documentation**

#### 3.8.5.1 **override int Length [get]**

Gets the length of the character string in characters.

Value: Number of characters.

Reimplemented from [Asn1Type](#).

## 3.9 Asn1Choice Class Reference

Inherits [Asn1Type](#).

### 3.9.1 Detailed Description

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

#### Public Member Functions

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

#### Protected Attributes

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

#### Properties

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

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 [Asn1Choice](#) ()

The default constructor initializes the choiceID and value.

### 3.9.3 Member Function Documentation

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

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

##### Parameters:

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

##### Returns:

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

### 3.9.3.2 virtual `Asn1Type GetElement ()` [virtual]

This method returns the element object.

#### Returns:

element data member.

### 3.9.3.3 override `int GetHashCode ()`

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

#### Returns:

A hash code for the current Object.

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

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

#### Parameters:

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

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

## 3.9.4 Member Data Documentation

### 3.9.4.1 [NonSerialized] internal `int choiceID` [protected]

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

### 3.9.4.2 [NonSerialized] internal `Asn1Type element` [protected]

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

## 3.9.5 Property Documentation

### 3.9.5.1 virtual `int ChoiceID` [get]

Gets the choice identifier.

Value: choice option identifier

### 3.9.5.2 abstract System.String ElemName [get]

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

Value: choice option name

## 3.10 Asn1ChoiceExt Class Reference

Inherits [Asn1OpenType](#).

### 3.10.1 Detailed Description

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

### Public Member Functions

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

### Public Attributes

- short [choiceIndex](#)

### 3.10.2 Member Function Documentation

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

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

#### Parameters:

*buffer* Decode message buffer object

Reimplemented from [Asn1OctetString](#).

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1OpenType](#).

### 3.10.2.3 override void Encode (Asn1PerOutputStream *outs*) [virtual]

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

#### Parameters:

*outs* PER Output Stream object

Reimplemented from [Asn1OpenType](#).

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating element is explicitly tagged

Reimplemented from [Asn1OpenType](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

Reimplemented from [Asn1OpenType](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

#### Returns:

Length of encoded component

Reimplemented from [Asn1OpenType](#).

## 3.10.3 Member Data Documentation

### 3.10.3.1 short [choiceIndex](#)

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

## 3.11 Asn1ConsVioException Class Reference

Inherits [Asn1Exception](#).

### 3.11.1 Detailed Description

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

#### Public Member Functions

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

### 3.11.2 Constructor & Destructor Documentation

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

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

#### Parameters:

*varname* Name of variable that violates constraint

*value* Value of variable

## 3.12 Asn1DecodeBuffer Class Reference

Inherits [Asn1MessageBuffer](#).

### 3.12.1 Detailed Description

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

#### Public Member Functions

- virtual void [AddCaptureBuffer](#) (System.IO.MemoryStream buffer)
- virtual void [AddNamedEventHandler](#) ([Asn1NamedEventHandler](#) handler)
- virtual void [Capture](#) (int nbytes)
- virtual long [DecodeIntValue](#) (int length, bool signExtend)
- virtual int[] [DecodeOIDContents](#) (int llen)
- virtual int[] [DecodeRelOIDContents](#) (int llen)
- override System.IO.Stream [GetInputStream](#) ()
- virtual void [HexDump](#) ()
- virtual void [InvokeCharacters](#) (System.String svalue)
- virtual void [InvokeEndElement](#) (System.String name, int index)
- virtual void [InvokeStartElement](#) (System.String name, int index)
- virtual void [Mark](#) ()
- virtual void [Read](#) (byte[] buffer)
- virtual void [Read](#) (byte[] buffer, int offset, int nbytes)
- virtual int [Read](#) ()
- abstract int [ReadByte](#) ()
- virtual void [RemoveCaptureBuffer](#) (System.IO.MemoryStream buffer)
- virtual void [Reset](#) ()
- virtual void [SetInputStream](#) (byte[] msgdata, int offset, int length)
- virtual long [Skip](#) (long nbytes)

#### Protected Member Functions

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

#### Protected Attributes

- internal int [mByteCount](#)

#### Properties

- virtual int [ByteCount](#) [get]
- virtual [Asn1DecodeBuffer EventHandlerList](#) [set]
- virtual short [TypeCode](#) [set]

## 3.12.2 Member Function Documentation

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

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

#### Parameters:

*buffer* Buffer into which captured bytes are to be stored

### 3.12.2.2 virtual void AddNamedEventHandler (Asn1NamedEventHandler *handler*) [virtual]

This method adds a named event handler to the named event handler list for this buffer.

#### Parameters:

*handler* [Asn1NamedEventHandler](#) object to be added

### 3.12.2.3 virtual void Capture (int *nbytes*) [virtual]

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

#### Parameters:

*nbytes* Number of bytes to capture

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

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

#### Parameters:

*length* Length of encoded contents

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

#### Returns:

Decoded long integer value

### 3.12.2.5 virtual int [] DecodeOIDContents (int *llen*) [virtual]

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

#### Parameters:

*llen* Length of encoded contents

#### Returns:

Decoded object identifier value

### 3.12.2.6 **virtual int [] DecodeRelOIDContents (int *llen*)** [virtual]

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

#### **Parameters:**

*llen* Length of encoded contents

#### **Returns:**

Decoded object identifier value

### 3.12.2.7 **override System.IO.Stream GetInputStream ()** [virtual]

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

#### **Returns:**

New input stream object containing encoded message

Implements [Asn1MessageBuffer](#).

### 3.12.2.8 **virtual void HexDump ()** [virtual]

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

### 3.12.2.9 **virtual internal void Init ()** [protected, virtual]

This method initializes the input stream for decoding.

### 3.12.2.10 **virtual void InvokeCharacters (System.String *svalue*)** [virtual]

This method is used by the event handling logic to invoke the 'characters' event handling method when message contents are parsed.

#### **Parameters:**

*svalue* Stringified representation of a parsed value field

### 3.12.2.11 **virtual void InvokeEndElement (System.String *name*, int *index*)** [virtual]

This method is used by the event handling logic to invoke the 'endElement' event handling method when parsing of an element within a message is completed.

#### **Parameters:**

*name* Name of the element

*index* Index of element if SEQUENCE OF or SET OF element

### 3.12.2.12 virtual void InvokeStartElement (System.String *name*, int *index*) [virtual]

This method is used by the event handling logic to invoke the 'StartElement' event handling method when parsing of an element within a message is started.

#### Parameters:

- name* Name of the element
- index* Index of element if SEQUENCE OF or SET OF element

### 3.12.2.13 virtual void Mark () [virtual]

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

### 3.12.2.14 virtual void Read (byte[] *buffer*) [virtual]

This version of the read method reads the number of bytes equal to the length of the given input buffer. Throws, Exception thrown by C# System.IO.Stream for I/O error, for Stream as input data

#### Parameters:

- buffer* the buffer into which the data is read

#### Exceptions:

- Asn1EndOfBufferException* Thrown if at end-of-stream

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

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

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

#### Parameters:

- buffer* the buffer into which the data is read
- offset* the start offset of the data
- nbytes* number of bytes to read

#### Exceptions:

- Asn1EndOfBufferException* Thrown if at end-of-stream

### 3.12.2.16 virtual int Read () [virtual]

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

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

**Returns:**

byte that was read from the input stream

**Exceptions:**

*Asn1EndOfBufferException* Thrown if at end-of-stream

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

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

**Returns:**

Next 8-bit byte value from input stream

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

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

**Parameters:**

*buffer* Buffer in which captured bytes stored

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

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

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

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

**Parameters:**

*msgdata* Byte array containing encoded message data

*offset* Starting offset of data in the byte array

*length* Length (in bytes) of the encoded data

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

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

**Parameters:**

*nbytes* Number of bytes to skip

**Returns:**

Skipped number of bytes

### 3.12.3 Member Data Documentation

#### 3.12.3.1 internal int `mByteCount` [protected]

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

### 3.12.4 Property Documentation

#### 3.12.4.1 virtual int `ByteCount` [get]

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

#### 3.12.4.2 virtual `Asn1DecodeBuffer EventHandlerList` [set]

Sets the event handler list in this object to be equal to that in the given Decode buffer object.

Value: Decode buffer object

#### 3.12.4.3 virtual short `TypeCode` [set]

Sets the internal type code to the given value. This is a code describing the last type parsed by the decoder.

Value: Type code (codes are defined in `Asn1Type.cs`). The codes correspond to the UNIVERSAL tag ID values for the built-in types.

## 3.13 Asn1DiscreteCharSet Class Reference

Inherits [Asn1CharSet](#).

### 3.13.1 Detailed Description

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

#### Public Member Functions

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

#### Properties

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

### 3.13.2 Constructor & Destructor Documentation

#### 3.13.2.1 [Asn1DiscreteCharSet](#) (System.String *charSet*)

This constructor sets the permitted alphabet character set

##### Parameters:

*charSet* Permitted alphabet character set

#### 3.13.2.2 [Asn1DiscreteCharSet](#) (int[] *charSet*)

This constructor sets the permitted alphabet character set

##### Parameters:

*charSet* Permitted alphabet character set

### 3.13.3 Member Function Documentation

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

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

##### Parameters:

*index* Index of character within the character set

##### Returns:

Character at given index

**Exceptions:**

*Asn1ConsVioException* Thrown if index not within define range

Implements [Asn1CharSet](#).

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

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

**Parameters:**

*charValue* Character value to search for

**Returns:**

Index of character

**Exceptions:**

*Asn1ConsVioException* thrown if 'charValue' not found in set

Implements [Asn1CharSet](#).

### 3.13.4 Property Documentation

**3.13.4.1 override int MaxValue [get]**

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

Reimplemented from [Asn1CharSet](#).

## 3.14 Asn1EncodeBuffer Class Reference

Inherits [Asn1MessageBuffer](#).

### 3.14.1 Detailed Description

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

#### Public Member Functions

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

#### Public Attributes

- const int [SIZE\\_INCREMENT](#) = 1024

#### Protected Member Functions

- virtual internal void [CheckSize](#) (int bytesRequired)
- virtual internal void [InitBuffer](#) (int sizeIncrement)

#### Protected Attributes

- internal int [mByteIndex](#)
- internal byte[] [mData](#)
- internal int [mSizeIncrement](#)

#### Properties

- abstract byte[] [MsgCopy](#) [get]
- abstract int [MsgLength](#) [get]

### 3.14.2 Member Function Documentation

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

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

**Parameters:**

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

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

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

**Parameters:**

*varName* Name of the Decoded ASN1 Type

**3.14.2.3 virtual internal void CheckSize (int bytesRequired) [protected, virtual]**

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.

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

This method copies multiple bytes to the encode buffer

**Parameters:**

*value* Array of bytes to copy to the encode buffer

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

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

**Parameters:**

*value* The byte value to copy

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

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

**Parameters:**

*outs* Output stream object reference

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

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

### 3.14.2.8 virtual internal void InitBuffer (int *sizeIncrement*) [protected, virtual]

This method will initialize this class member variables.

#### Parameters:

*sizeIncrement* Buffer size increment in bytes

### 3.14.2.9 abstract void Reset () [pure virtual]

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

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

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

#### Parameters:

*outs* Output stream to which record is to be written

## 3.14.3 Member Data Documentation

### 3.14.3.1 internal int **mByteIndex** [protected]

This variable holds the position of the byte array for encode buffer.

### 3.14.3.2 internal byte [] **mData** [protected]

This variable holds the encoded data as byte array.

### 3.14.3.3 internal int **mSizeIncrement** [protected]

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

### 3.14.3.4 const int **SIZE\_INCREMENT** = 1024

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

## 3.14.4 Property Documentation

### 3.14.4.1 abstract byte [] **MsgCopy** [get]

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

Value: byte array containing encoded message

#### 3.14.4.2 **abstract int MsgLength** [get]

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

Value: length of encoded message component

## 3.15 Asn1EndOfBufferException Class Reference

Inherits [Asn1Exception](#).

### 3.15.1 Detailed Description

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

#### Public Member Functions

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

### 3.15.2 Constructor & Destructor Documentation

#### 3.15.2.1 [Asn1EndOfBufferException](#) ([Asn1DecodeBuffer](#) *buffer*)

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

#### Parameters:

*buffer* Decode buffer object reference

## 3.16 Asn1Enumerated Class Reference

Inherits [Asn1Type](#).

### 3.16.1 Detailed Description

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

#### Public Member Functions

- [Asn1Enumerated](#) (int value\_)
- [Asn1Enumerated](#) ()
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XerEncodeBuffer buffer)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (int value)
- override int [GetHashCode](#) ()
- virtual int [ParseValue](#) (System.String value)
- override System.String [ToString](#) ()

#### Public Attributes

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

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.16.2 Constructor & Destructor Documentation

#### 3.16.2.1 [Asn1Enumerated](#) ()

The default constructor sets the enumerated value to undefined.

### 3.16.2.2 [AsnEnumerated](#) (int *value\_*)

This constructor creates an enumerated object from a integer value.

#### Parameters:

*value\_* Integer value

## 3.16.3 Member Function Documentation

### 3.16.3.1 virtual void [Decode](#) ([AsnPerDecodeBuffer](#) *buffer*, long *lower*, long *upper*) [virtual]

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

#### Parameters:

*buffer* PER Decode message buffer object

*lower* Smallest enumerated value in the set

*upper* Largest enumerated value in the set

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [AsnType](#).

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

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

#### Parameters:

*buffer* String containing data to be decoded. The value is assumed to be clear text with wrapping element delimiters (i.e. "&lt;" and "&gt;") removed.

*attrs* Attributes string from element tag

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### Parameters:

*outs* PER Output Stream object

*lower* Smallest enumerated value in the set

*upper* Largest enumerated value in the set

#### Exceptions:

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

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1Type](#).

### 3.16.3.8 virtual void Encode (Asn1XerEncodeBuffer *buffer*) [virtual]

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

#### Parameters:

*buffer* Encode message buffer object

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

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

#### Parameters:

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* PER Encode message buffer object

*lower* Smallest enumerated value in the set

*upper* Largest enumerated value in the set

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length of component or negative status value

Reimplemented from [Asn1Type](#).

### 3.16.3.12 **override bool Equals (System.Object value)**

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

#### **Parameters:**

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

#### **Returns:**

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

### 3.16.3.13 **virtual bool Equals (int value) [virtual]**

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

#### **Parameters:**

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

#### **Returns:**

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

### 3.16.3.14 **override int GetHashCode ()**

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

#### **Returns:**

A hash code for the current Object.

### 3.16.3.15 **virtual int ParseValue (System.String value) [virtual]**

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

#### **Parameters:**

*value* enumeration text

#### **Returns:**

Stringified representation of the value

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

### 3.16.3.16 **override System.String ToString ()**

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

#### **Returns:**

Stringified representation of the value

## 3.16.4 Member Data Documentation

### 3.16.4.1 new readonly [Asn1Tag\\_TAG](#) [static]

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

Reimplemented from [Asn1Type](#).

### 3.16.4.2 [NonSerialized] int [mValue](#)

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

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

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

## 3.17 Asn1Exception Class Reference

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

### 3.17.1 Detailed Description

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

### Public Member Functions

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

### 3.17.2 Constructor & Destructor Documentation

#### 3.17.2.1 [Asn1Exception](#) (System.String *message*)

This constructor passes the given message text to the superclass.

#### Parameters:

*message* Error message text

#### 3.17.2.2 [Asn1Exception](#) (System.String *message*, System.Exception *innerException*)

This constructor passes the given message text to the superclass.

#### Parameters:

*message* Error message text

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

#### 3.17.2.3 [Asn1Exception](#) ([Asn1DecodeBuffer](#) *buffer*, System.String *message*)

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

#### Parameters:

*buffer* ASN.1 Decode buffer object reference

*message* Error message text

## 3.18 Asn1GeneralizedTime Class Reference

Inherits [Asn1Time](#).

### 3.18.1 Detailed Description

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

#### Public Member Functions

- [Asn1GeneralizedTime](#) (System.String data, bool useDerRules)
- [Asn1GeneralizedTime](#) (System.String data)
- [Asn1GeneralizedTime](#) (bool useDerRules)
- [Asn1GeneralizedTime](#) ()
- override System.Int32 [CompareTo](#) (System.Object obj)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- virtual void [EncodeXMLData](#) (Asn1XmlXerEncoder buffer)
- override void [ParseString](#) (System.String data)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

#### Protected Member Functions

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

#### Properties

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

### 3.18.2 Constructor & Destructor Documentation

#### 3.18.2.1 [Asn1GeneralizedTime](#) ()

The default constructor creates an empty time string object.

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

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

#### Parameters:

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

### 3.18.2.3 [Asn1GeneralizedTime](#) (System.String *data*)

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

#### Parameters:

*data* Character string

### 3.18.2.4 [Asn1GeneralizedTime](#) (System.String *data*, bool *useDerRules*)

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

#### Parameters:

*data* Character string

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

## 3.18.3 Member Function Documentation

### 3.18.3.1 override System.Int32 [CompareTo](#) (System.Object *obj*) [virtual]

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

#### Parameters:

*obj* the Object to be compared.

#### Returns:

The difference in Ticks with the specified object.

Reimplemented from [Asn1Time](#).

### 3.18.3.2 internal override bool [CompileString](#) () [protected, virtual]

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

#### Returns:

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

Implements [Asn1Time](#).

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

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

**Parameters:**

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

Reimplemented from [Asn1Type](#).

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

This method encodes ASN.1 Generalized Time string into xsd:dateTime format with element and namespace prefix tag according to the XML Encoding as specified in the XML schema standard(asn2xsd).

**Parameters:**

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

Reimplemented from [Asn1CharString](#).

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

This method encodes ASN.1 Generalized Time string into xsd:dateTime format with element name tag according to the XML Encoding as specified in the XER itu-t standard.

**Parameters:**

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

Reimplemented from [Asn1CharString](#).

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

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

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

**Parameters:**

- outs* BER Output Stream object
- explicitTagging* Flag indicating explicit tagging should be done

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

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

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

### 3.18.3.8 virtual void EncodeXMLData (Asn1XmlXerEncoder *buffer*) [virtual]

This method creates XML TimeString from the Asn1 GeneralizedTime String. XML date-Time format YYYY-MM-DDTHH:MM:SS[.SSSS][Z|(+-) HH:MM] Asn1 GeneralizedTime format YYYYMMDDHH[MM[SS[(.|.)SSSS]]][Z|(+-)HH(MM))

#### Parameters:

- buffer* String buffer to hold the converted xml time string

#### Exceptions:

- [Asn1Exception](#) any exception exist in Asn.1 Generalized Time string

### 3.18.3.9 override void ParseString (System.String *data*) [virtual]

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

#### Parameters:

- data* The time string value to be parsed.

#### Exceptions:

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

Implements [Asn1Time](#).

## 3.18.4 Member Data Documentation

### 3.18.4.1 new readonly Asn1Tag \_TAG [static]

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

Reimplemented from [Asn1Type](#).

### 3.18.5 Property Documentation

#### 3.18.5.1 virtual int Century [get, set]

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

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

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

## 3.19 Asn1GeneralString Class Reference

Inherits [Asn1VarWidthCharString](#).

### 3.19.1 Detailed Description

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

#### Public Member Functions

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

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.19.2 Constructor & Destructor Documentation

#### 3.19.2.1 [Asn1GeneralString](#) ()

The default constructor creates an empty string object.

#### 3.19.2.2 [Asn1GeneralString](#) (System.String data)

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

#### Parameters:

*data* Character string

### 3.19.3 Member Function Documentation

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.19.4 Member Data Documentation

### 3.19.4.1 new readonly Asn1Tag \_TAG [static]

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

Reimplemented from [Asn1Type](#).

## 3.20 Asn1GraphicString Class Reference

Inherits [Asn1VarWidthCharString](#).

### 3.20.1 Detailed Description

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

#### Public Member Functions

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

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.20.2 Constructor & Destructor Documentation

#### 3.20.2.1 [Asn1GraphicString](#) ()

The default constructor creates an empty string object.

#### 3.20.2.2 [Asn1GraphicString](#) (System.String data)

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

#### Parameters:

*data* string representation of GraphicString

### 3.20.3 Member Function Documentation

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.20.4 Member Data Documentation

### 3.20.4.1 new readonly Asn1Tag \_TAG [static]

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

Reimplemented from [Asn1Type](#).

## 3.21 Asn1IA5String Class Reference

Inherits [Asn18BitCharString](#).

### 3.21.1 Detailed Description

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

#### Public Member Functions

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

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.21.2 Constructor & Destructor Documentation

#### 3.21.2.1 [Asn1IA5String](#) ()

The default constructor creates an empty string object.

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

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

#### Parameters:

*data* string representation of IA5String

### 3.21.3 Member Function Documentation

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.21.4 Member Data Documentation

### 3.21.4.1 new readonly [Asn1Tag\\_TAG](#) [static]

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

Reimplemented from [Asn1Type](#).

## 3.22 Asn1InputStream Interface Reference

### 3.22.1 Detailed Description

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

#### Public Member Functions

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

### 3.22.2 Member Function Documentation

#### 3.22.2.1 int Available ()

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

##### Returns:

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

##### Exceptions:

*System.SystemException* if an I/O error occurs.

#### 3.22.2.2 void Close ()

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

##### Exceptions:

*System.SystemException* if an I/O error occurs.

#### 3.22.2.3 void Mark ()

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

#### 3.22.2.4 bool MarkSupported ()

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

##### Returns:

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

### 3.22.2.5 void Reset ()

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

### 3.22.2.6 long Skip (long *nbytes*)

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

#### Parameters:

*nbytes* Number of bytes to skip

#### Exceptions:

*System.SystemException* if an I/O error occurs.

#### Returns:

Skipped number of bytes

## 3.23 Asn1Integer Class Reference

Inherits [Asn1Type](#).

### 3.23.1 Detailed Description

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

#### Public Member Functions

- [Asn1Integer](#) (long value)
- [Asn1Integer](#) ()
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, Object lower, Object upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, Object upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, Object lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, Object lower, Object upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, Object upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, Object lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, Object lower, Object upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, Object upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, Object lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (long value)
- virtual int [GetBitCount](#) ()
- override int [GetHashCode](#) ()
- virtual int [GetUnsignedBitCount](#) ()
- override System.String [ToString](#) ()

#### Static Public Member Functions

- static int [GetBitCount](#) (long ivalue)
- static int [GetUnsignedBitCount](#) (long ivalue)

## Public Attributes

- [NonSerialized] long [mValue](#)

## Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

## 3.23.2 Constructor & Destructor Documentation

### 3.23.2.1 [Asn1Integer](#) ()

The default constructor sets the integer value to zero.

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

This constructor creates an integer object from a integer value.

#### Parameters:

*value* Integer value

## 3.23.3 Member Function Documentation

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

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

#### Parameters:

*buffer* PER Decode message buffer object

*lower* Lower bound equal MIN

*upper* Upper bound equal MAX

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

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

#### Parameters:

*buffer* PER Decode message buffer object

*lower* Lower bound of the integer range

*upper* Upper bound equal MAX

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

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

#### Parameters:

*buffer* PER Decode message buffer object

*lower* Lower bound equal MIN

*upper* Upper bound of the integer range

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

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

#### Parameters:

*buffer* PER Decode message buffer object

*lower* Lower bound of the integer range

*upper* Upper bound of the integer range

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

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

#### Parameters:

*buffer* PER Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

### 3.23.3.9 virtual void Encode (Asn1PerOutputStream *outs*, Object *lower*, Object *upper*) [virtual]

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

#### Parameters:

*outs* PER Encode message buffer object

*lower* Lower bound equal MIN

*upper* Upper bound equal MAX

### 3.23.3.10 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, Object *upper*) [virtual]

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

#### Parameters:

*outs* PER Encode message buffer object

*lower* Lower bound (inclusive) of integer being encoded

*upper* Upper bound equal MAX

### 3.23.3.11 virtual void Encode (Asn1PerOutputStream *outs*, Object *lower*, long *upper*) [virtual]

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

#### Parameters:

*outs* PER Encode message buffer object

*lower* Lower bound equal MIN

*upper* Upper bound (inclusive) of integer being encoded

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

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### Parameters:

*outs* PER Encode message buffer object

*lower* Lower bound (inclusive) of integer being encoded

*upper* Upper bound (inclusive) of integer being encoded

#### Exceptions:

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

### 3.23.3.13 override void Encode (Asn1PerOutputStream outs) [virtual]

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### Parameters:

*outs* PER Encode message buffer object

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1Type](#).

**3.23.3.17** **virtual void Encode (Asn1PerEncodeBuffer *buffer*, Object *lower*, Object *upper*)** [virtual]

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

**Parameters:**

*buffer* PER Encode message buffer object

*lower* Lower bound equal MIN

*upper* Upper bound equal MAX

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

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

**Parameters:**

*buffer* PER Encode message buffer object

*lower* Lower bound (inclusive) of integer being encoded

*upper* Upper bound equal MAX

### 3.23.3.19 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Object *lower*, long *upper*) [virtual]

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

#### Parameters:

- buffer* PER Encode message buffer object
- lower* Lower bound equal MIN
- upper* Upper bound (inclusive) of integer being encoded

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

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

#### Parameters:

- buffer* PER Encode message buffer object
- lower* Lower bound (inclusive) of integer being encoded
- upper* Upper bound (inclusive) of integer being encoded

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

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

#### Parameters:

- buffer* PER Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

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

#### Returns:

Length of component or negative status value

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*attribute* Element attribute value

Reimplemented from [Asn1Type](#).

### 3.23.3.24 **override bool Equals (System.Object *value*)**

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

#### **Parameters:**

*value* The Object to compare with the current Object. Object should be instance of [Asn1Integer](#).

#### **Returns:**

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

### 3.23.3.25 **virtual bool Equals (long *value*)** [virtual]

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

#### **Parameters:**

*value* The long value to compare with the current Object.

#### **Returns:**

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

### 3.23.3.26 **virtual int GetBitCount ()** [virtual]

This method calculates the count of bits in the contained integer value.

#### **Returns:**

Bit count.

### 3.23.3.27 **static int GetBitCount (long *ivalue*)** [static]

This method calculates the count of bits in an integer value.

#### **Parameters:**

*ivalue* Integer value in which to count bits.

**Returns:**

Bit count.

**3.23.3.28 override int GetHashCode ()**

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

**Returns:**

A hash code for the current Object.

**3.23.3.29 virtual int GetUnsignedBitCount () [virtual]**

This method calculates the count of bits in the contained unsigned integer value.

**Returns:**

Bit count.

**3.23.3.30 static int GetUnsignedBitCount (long *ivalue*) [static]**

This method calculates the count of bits in an unsigned integer value.

**Parameters:**

*ivalue* Integer value in which to count bits.

**Returns:**

Bit count.

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

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

**Returns:**

Stringified representation of the value

**3.23.4 Member Data Documentation****3.23.4.1 new readonly [Asn1Tag\\_TAG](#) [static]**

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

Reimplemented from [Asn1Type](#).

#### 3.23.4.2 [NonSerialized] long **mValue**

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

## 3.24 Asn1InvalidArgException Class Reference

Inherits [Asn1Exception](#).

### 3.24.1 Detailed Description

This class defines the 'ASN.1 invalid argument' exception that is thrown when an argument that is passed to a method is determined to be invalid (for example, not within a defined range)..

### Public Member Functions

- [Asn1InvalidArgException](#) (System.String argName, System.String argValue)

### 3.24.2 Constructor & Destructor Documentation

#### 3.24.2.1 [Asn1InvalidArgException](#) (System.String *argName*, System.String *argValue*)

This constructor creates an exception object with a textual message describing the argument that was invalid..

#### Parameters:

*argName* Name of invalid argument

*argValue* Value of invalid argument

## 3.25 Asn1InvalidChoiceOptionException Class Reference

Inherits [Asn1Exception](#).

### 3.25.1 Detailed Description

This class defines the 'ASN.1 invalid choice option' exception that is thrown when a CHOICE construct is detected to contain an element that is not within the given set.

#### Public Member Functions

- [Asn1InvalidChoiceOptionException](#) ()
- [Asn1InvalidChoiceOptionException](#) (Asn1PerDecodeBuffer buffer, int index)
- [Asn1InvalidChoiceOptionException](#) (Asn1BerDecodeBuffer buffer, [Asn1Tag](#) tag)

### 3.25.2 Constructor & Destructor Documentation

#### 3.25.2.1 [Asn1InvalidChoiceOptionException](#) (Asn1BerDecodeBuffer *buffer*, [Asn1Tag](#) *tag*)

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

##### Parameters:

*buffer* BER Decode buffer object reference

*tag* Tag value of duplicate element

#### 3.25.2.2 [Asn1InvalidChoiceOptionException](#) (Asn1PerDecodeBuffer *buffer*, int *index*)

This constructor creates an exception object with a textual message describing the PER choice index of the invalid element..

##### Parameters:

*buffer* PER Decode buffer object reference

*index* Parsed choice index value

#### 3.25.2.3 [Asn1InvalidChoiceOptionException](#) ()

The default constructor is invoked in the encode logic if the object assigned to the choice item is not in the allowed set..

## 3.26 Asn1InvalidEnumException Class Reference

Inherits [Asn1Exception](#).

### 3.26.1 Detailed Description

This class defines the 'ASN.1 invalid enum' exception that is thrown when an enumerated value is not within the defined set of values.

### Public Member Functions

- [Asn1InvalidEnumException](#) (System.String data)
- [Asn1InvalidEnumException](#) (long data)

### 3.26.2 Constructor & Destructor Documentation

#### 3.26.2.1 [Asn1InvalidEnumException](#) (long data)

This constructor creates an exception object with a default textual message with integer value.

#### Parameters:

*data* Invalid enumerated value

#### 3.26.2.2 [Asn1InvalidEnumException](#) (System.String data)

This constructor creates an exception object with a default textual message with textual enumerated value.

#### Parameters:

*data* Invalid enumerated value

## 3.27 Asn1InvalidLengthException Class Reference

Inherits [Asn1Exception](#).

### 3.27.1 Detailed Description

This class defines the 'ASN.1 invalid length' exception that is thrown when a length is determined to be invalid.

Things that can cause this to be thrown are:

- Constructor length field is not sum of parts.
- Object identifier length is not sum of sub ID lengths.

### Public Member Functions

- [Asn1InvalidLengthException \(\)](#)

### 3.27.2 Constructor & Destructor Documentation

#### 3.27.2.1 [Asn1InvalidLengthException \(\)](#)

This constructor creates an exception object with a default textual message.

## 3.28 Asn1InvalidObjectIDException Class Reference

Inherits [Asn1Exception](#).

### 3.28.1 Detailed Description

This class defines the 'ASN.1 invalid object identifier' exception that is thrown when an Object Identifier is determined to be invalid.

Things that can cause this to be thrown are:

- Max subID's (128) exceeded.
- Not enough subID's in the OID (must contain at least 2).
- First subID > 2 and/or second subID > 40.

### Public Member Functions

- [Asn1InvalidObjectIDException \(\)](#)

### 3.28.2 Constructor & Destructor Documentation

#### 3.28.2.1 [Asn1InvalidObjectIDException \(\)](#)

This constructor creates an exception object with a default textual message.

## 3.29 Asn1MessageBuffer Class Reference

Inherited by [Asn1DecodeBuffer](#), and [Asn1EncodeBuffer](#).

### 3.29.1 Detailed Description

This is the base class for all of the different message buffer types. This includes the BER and PER encode and decode message buffer classes.

#### Public Member Functions

- abstract `System.IO.Stream` [GetInputStream](#) ()
- void [SetKey](#) (byte[] rtkey)

#### Static Public Member Functions

- static void [HexDump](#) (`System.IO.Stream` ins)
- static void [HexDump](#) (`System.IO.Stream` ins, `System.IO.StreamWriter` outs)

### 3.29.2 Member Function Documentation

#### 3.29.2.1 abstract `System.IO.Stream` [GetInputStream](#) () [pure virtual]

This abstract method must be implemented by all of the derived classes. It returns an input stream object reference to the message buffer contents (i.e. the encoded data).

##### Returns:

Input stream object reference

Implemented in [Asn1DecodeBuffer](#).

#### 3.29.2.2 static void [HexDump](#) (`System.IO.Stream` *ins*) [static]

This method prints a formatted hex dump of the contents of the given input stream to the standard output stream.

##### Parameters:

*ins* `System.IO.Stream` containg data to be dumped

#### 3.29.2.3 static void [HexDump](#) (`System.IO.Stream` *ins*, `System.IO.StreamWriter` *outs*) [static]

This method prints a formatted hex dump of the contents of the given input stream to the given output stream.

##### Parameters:

*ins* `System.IO.Stream` containg data to be dumped

*outs* `StreamWriter` to which formatted data is to be written

#### 3.29.2.4 void SetKey (byte[] *rtkey*)

This method is used with the limited run-time to set a run-time key value generated by the compiler to allow the run-time to operate on the licensed hosts. This is not used in the unlimited redistribution versions.

#### Parameters:

*rtkey* Run-time key generated by ASN1C

## 3.30 Asn1MissingRequiredException Class Reference

Inherits [Asn1Exception](#).

### 3.30.1 Detailed Description

This class defines the 'ASN.1 set missing required element' exception that is thrown from Decode methods when a SET construct is decoded and found to be missing a required element..

#### Public Member Functions

- [Asn1MissingRequiredException](#) (System.String elemName)
- [Asn1MissingRequiredException](#) (Asn1BerDecodeBuffer buffer)

### 3.30.2 Constructor & Destructor Documentation

#### 3.30.2.1 [Asn1MissingRequiredException](#) (Asn1BerDecodeBuffer *buffer*)

This constructor creates an exception object with a textual message describing the error.

##### Parameters:

*buffer* BER decode buffer object reference

#### 3.30.2.2 [Asn1MissingRequiredException](#) (System.String *elemName*)

This constructor creates an exception object with a textual message describing the error including the name of the required element that is missing.

##### Parameters:

*elemName* Name of missing required element

## 3.31 Asn1NamedEventHandler Interface Reference

Inherited by [Asn1TraceHandler](#).

### 3.31.1 Detailed Description

This interface defines the methods that must be implemented to define a SAX-like event handler. These methods are invoked from within the generated C# decode logic when significant events occur during the parsing of an ASN.1 message.

#### Public Member Functions

- void [Characters](#) (System.String svalue, short typeCode)
- void [EndElement](#) (System.String name, int index)
- void [StartElement](#) (System.String name, int index)

### 3.31.2 Member Function Documentation

#### 3.31.2.1 void Characters (System.String svalue, short typeCode)

The Characters callback method is invoked when content (primitive data) is encountered. A stringified representation of the parsed value is returned.

##### Parameters:

*svalue* Stringified representation of the parsed value. The representation will be in ASN.1 value format.

*typeCode* Identifier specifying the type of the parsed data variable. The enumerated list of values that might appear here is provided in the the [Asn1Type](#) class (see the documentation on this class for a full list of the names).

##### See also:

<seealso cref=Asn1Type The type codes are member of [Asn1Type](#) class

Implemented in [Asn1TraceHandler](#).

#### 3.31.2.2 void EndElement (System.String name, int index)

The EndElement callback method is invoked when the end of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is detected.

##### Parameters:

*name* Name of the parsed element.

*index* Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implemented in [Asn1TraceHandler](#).

### 3.31.2.3 void StartElement (System.String *name*, int *index*)

The StartElement callback method is invoked when the start of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is encountered.

#### Parameters:

*name* Name of the parsed element.

*index* Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implemented in [Asn1TraceHandler](#).

## 3.32 Asn1Null Class Reference

Inherits [Asn1Type](#).

### 3.32.1 Detailed Description

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

#### Public Member Functions

- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (object o)
- override System.String [ToString](#) ()

#### Static Public Attributes

- static new readonly [Asn1Tag \\_TAG](#)
- static readonly [Asn1Null NULL\\_VALUE](#) = new [Asn1Null](#)()

### 3.32.2 Member Function Documentation

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

This method decodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

##### Parameters:

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

##### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

### 3.32.2.5 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1Type](#).

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

This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

**Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

**Returns:**

Length (in octets) of encoded component

Reimplemented from [Asn1Type](#).

### 3.32.2.11 override bool Equals (object o)

Tests for equality with any other object. Returns true if the input type is an [Asn1Null](#) object and false otherwise.

### 3.32.2.12 override System.String ToString ()

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

#### Returns:

Stringified representation of the value

## 3.32.3 Member Data Documentation

### 3.32.3.1 new readonly [Asn1Tag \\_TAG](#) [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 5).

Reimplemented from [Asn1Type](#).

### 3.32.3.2 readonly [Asn1Null NULL\\_VALUE](#) = new [Asn1Null\(\)](#) [static]

The `NULL_VALUE` constant represents a NULL value.

## 3.33 Asn1NumericString Class Reference

Inherits [Asn18BitCharString](#).

### 3.33.1 Detailed Description

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

#### Public Member Functions

- [Asn1NumericString](#) (System.String data)
- [Asn1NumericString](#) ()
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)
- static readonly [Asn1CharSet\\_CHARSET](#)

### 3.33.2 Constructor & Destructor Documentation

#### 3.33.2.1 [Asn1NumericString](#) ()

The default constructor creates an empty string object.

#### 3.33.2.2 [Asn1NumericString](#) (System.String data)

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

#### Parameters:

*data* string representation of numeric string

### 3.33.3 Member Function Documentation

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

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

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

**Parameters:**

- buffer* Decode message buffer object
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

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

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

**Parameters:**

- buffer* Decode message buffer object

Reimplemented from [Asn18BitCharString](#).

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

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

**Parameters:**

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

Reimplemented from [Asn1Type](#).

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

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

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

**Parameters:**

- outs* BER Output Stream object
- explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

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

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*buffer* Encode message buffer object

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

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

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

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

#### Parameters:

*buffer* Encode message buffer object

Reimplemented from [Asn18BitCharString](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.33.4 Member Data Documentation

### 3.33.4.1 new readonly Asn1Tag \_TAG [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 18).

Reimplemented from [Asn1Type](#).

### 3.33.4.2 readonly `Asn1CharSet CHARSET` [static]

#### Initial value:

```
new Asn1DiscreteCharSet (" 0123456789")
```

The `CHARSET` constant specifies the character set for a numeric string in canonical order.

## 3.34 Asn1ObjectDescriptor Class Reference

Inherits [Asn1VarWidthCharString](#).

### 3.34.1 Detailed Description

This is a container class for holding the components of an ASN.1 object descriptor value.

#### Public Member Functions

- [Asn1ObjectDescriptor](#) (System.String data)
- [Asn1ObjectDescriptor](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.34.2 Constructor & Destructor Documentation

#### 3.34.2.1 [Asn1ObjectDescriptor](#) ()

The default constructor creates an empty string object.

#### 3.34.2.2 [Asn1ObjectDescriptor](#) (System.String data)

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

#### Parameters:

*data* string representation of ObjectDescriptor

### 3.34.3 Member Function Documentation

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.34.4 Member Data Documentation

### 3.34.4.1 new readonly [Asn1Tag\\_TAG](#) [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 7).

Reimplemented from [Asn1Type](#).

## 3.35 Asn1ObjectIdentifier Class Reference

Inherits [Asn1Type](#).

Inherited by [Asn1RelativeOID](#).

### 3.35.1 Detailed Description

This is a container class for holding the components of an ASN.1 object identifier value.

#### Public Member Functions

- virtual void [Append](#) (int[] value2)
- [Asn1ObjectIdentifier](#) (int[] value)
- [Asn1ObjectIdentifier](#) ()
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override bool [Equals](#) (System.Object value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

#### Public Attributes

- const int [MAXSUBIDS](#) = 128
- [NonSerialized] int[] [mValue](#)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.35.2 Constructor & Destructor Documentation

#### 3.35.2.1 [Asn1ObjectIdentifier](#) ()

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

#### 3.35.2.2 [Asn1ObjectIdentifier](#) (int[] value)

This constructor initializes the object identifier from the given array of integer subidentifier values.

#### Parameters:

*value* Array of subidentifiers

### 3.35.3 Member Function Documentation

#### 3.35.3.1 virtual void Append (int[] *value2*) [virtual]

This method appends an object identifier value onto the existing value. A typical use of this method would be for SNMP objects to create the base and index parts.

**Parameters:**

*value2* Array of subidentifiers to append

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

This method decodes an ASN.1 object identifier value using the packed encoding rules (PER).

**Parameters:**

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

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

**Parameters:**

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

Reimplemented in [Asn1RelativeOID](#).

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

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

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

Reimplemented in [Asn1RelativeOID](#).

### 3.35.3.6 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes an ASN.1 object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

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

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

This method encodes an ASN.1 object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

**Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

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

This method encodes an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

**Parameters:**

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

**Returns:**

Length of encoded component in octets

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

### 3.35.3.12 **override bool Equals (System.Object value)**

This method compares this object identifier to the given one for equality.

#### **Parameters:**

*value* The Object to compare with the current Object. Object should be instance of [Asn1ObjectIdentifier](#).

#### **Returns:**

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

### 3.35.3.13 **override int GetHashCode ()**

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

#### **Returns:**

A hash code for the current Object.

### 3.35.3.14 **override System.String ToString ()**

This method will return a string representation of the OID value. The format is the ASN.1 value format for this type. Note that textual subidentifiers are not used, only numeric values..

#### **Returns:**

Stringified representation of the value

## 3.35.4 **Member Data Documentation**

### 3.35.4.1 **new readonly Asn1Tag \_TAG [static]**

The \_TAG constant describes the universal tag for this data type (UNIVERSAL 6).

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1RelativeOID](#).

### 3.35.4.2 **const int MAXSUBIDS = 128**

The MAXSUBIDS constant specifies the maximum number of subidentifiers that can appear in an OID value (128).

### 3.35.4.3 **[NonSerialized] int [] mValue**

The mValue public member variable is where the object identifier value is stored.

## 3.36 Asn1OctetString Class Reference

Inherits [Asn1Type](#).

Inherited by [Asn1OpenType](#).

### 3.36.1 Detailed Description

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

#### Public Member Functions

- [Asn1OctetString](#) (System.String value)
- [Asn1OctetString](#) (byte[] data, int offset, int nbytes)
- [Asn1OctetString](#) (byte[] data)
- [Asn1OctetString](#) ()
- virtual int [CompareTo](#) (System.Object octstr)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- override bool [Equals](#) (System.Object value)
- bool [Equals](#) (byte[] value)
- override int [GetHashCode](#) ()
- virtual System.IO.Stream [toInputStream](#) ()
- override System.String [ToString](#) ()

#### Public Attributes

- [NonSerialized] byte[] [mValue](#)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

#### Properties

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

## 3.36.2 Constructor & Destructor Documentation

### 3.36.2.1 [AsnOctetString \(\)](#)

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

### 3.36.2.2 [AsnOctetString \(byte\[\] data\)](#)

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

#### Parameters:

*data* Byte array containing an octet string in binary form.

### 3.36.2.3 [AsnOctetString \(byte\[\] data, int offset, int nbytes\)](#)

This constructor initializes an octet string from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

#### Parameters:

*data* Byte array containing an octet string in binary form.

*offset* The offset in array at which to begin copy.

*nbytes* Number of bytes to copy from target array

### 3.36.2.4 [AsnOctetString \(System.String value\)](#)

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

Examples of valid value formats are as follows:

Binary string: '11010010111001'B

Hex string: '0fa56920014abc'H

Char string: 'abcdefg'

#### Parameters:

*value* The ASN.1 value specification text

## 3.36.3 Member Function Documentation

### 3.36.3.1 [virtual int CompareTo \(System.Object octstr\) \[virtual\]](#)

This method compares two [AsnOctetString](#) objects for equality. The OCTET STRING's are equal if a) all octets are equal, and b) the lengths are the same.

This method is required to implement the Comparable interface used for sorting.

**Parameters:**

*octstr* [Asn1OctetString](#) to compare

**Returns:**

0 if equal, 1 if this string is greater than supplied string, -1 if this string is less than supplied string.

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

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

**Parameters:**

*buffer* Decode message buffer object

*lower* Lower bound (inclusive) of size constraint

*upper* Upper bound (inclusive) of size constraint

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

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

**Parameters:**

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#).

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

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

**Parameters:**

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

This method decodes an ASN.1 octet string type using the XML schema encoding rules(asn2xsd).

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### Parameters:

*outs* PER Output Stream object

*lower* Lower bound (inclusive) of size constraint

*upper* Upper bound (inclusive) of size constraint

#### Exceptions:

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

### 3.36.3.8 override void Encode (Asn1PerOutputStream *outs*) [virtual]

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

Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

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

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

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

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

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

**Parameters:**

*outs* BER Output Stream object  
*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

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

This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).

**Parameters:**

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

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

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

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

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

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

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

#### Parameters:

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length of encoded component

Reimplemented from [Asn1Type](#).

Reimplemented in [Asn1ChoiceExt](#), and [Asn1OpenType](#).

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

This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).

#### Parameters:

*buffer* Encode message buffer object

*elemName* XML element name used to wrap string

*attribute* Element attribute value

Reimplemented from [Asn1Type](#).

### 3.36.3.16 override bool Equals (System.Object *value*)

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

#### Parameters:

*value* The Object to compare with the current Object. Object should be instance of [Asn1OctetString](#).

#### Returns:

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

### 3.36.3.17 **bool Equals (byte[] value)**

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

#### **Parameters:**

*value* The byte array to compare with the current Object.

#### **Returns:**

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

### 3.36.3.18 **override int GetHashCode ()**

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

#### **Returns:**

A hash code for the current Object.

### 3.36.3.19 **virtual System.IO.Stream toInputStream () [virtual]**

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

#### **Returns:**

Reference to `System.IO.MemoryStream` object

### 3.36.3.20 **override System.String ToString ()**

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

#### **Returns:**

Stringified representation of the value

Reimplemented in [Asn1OpenType](#).

## 3.36.4 **Member Data Documentation**

### 3.36.4.1 **new readonly Asn1Tag \_TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 4).

Reimplemented from [Asn1Type](#).

### 3.36.4.2 **[NonSerialized] byte [] mValue**

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

### 3.36.5 Property Documentation

#### 3.36.5.1 `override int Length` [get]

Gets the length of the OCTET STRING in octets.

Value: length of the octet string

Reimplemented from [Asn1Type](#).

## 3.37 Asn1OpenExt Class Reference

Inherits [Asn1Type](#).

### 3.37.1 Detailed Description

This is a container class for holding open type elements that may occur within an open type extension (i.e. a ... at the end of a constructed type or a ..., ... at some other point in a constructed type).

### Public Member Functions

- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeComponent](#) (Asn1BerDecodeBuffer buffer)
- virtual void [DecodeEventComponent](#) (Asn1BerDecodeBuffer buffer)
- virtual [Asn1OpenType DecodeOpenType](#) (Asn1PerDecodeBuffer buffer, bool present, int index)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XerEncoder buffer)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- virtual void [EncodeExtBits](#) (Asn1PerEncodeBuffer buffer)
- virtual void [SetOpenType](#) ([Asn1OpenType](#) obj, int index)
- virtual void [ShrinkArray](#) (int numrecs)
- override System.String [ToString](#) ()

### Public Attributes

- [NonSerialized] System.Collections.ArrayList [mValue](#)

### 3.37.2 Member Function Documentation

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

This method decodes an open type extension in a SEQUENCE or SET construct using the packed encoding rules (PER). This method will capture each extension item in a separate open type object and store it in the `mValue` public member list variable. If optional items are absent, null placeholders will be inserted in the list.

#### Parameters:

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

- buffer* Decode message buffer object
- explicitTagging* Flag indicating element is explicitly tagged
- implicitLength* Length if implicit element

Reimplemented from [Asn1Type](#).

**3.37.2.3 virtual void DecodeComponent (Asn1BerDecodeBuffer *buffer*)** [virtual]

This method decodes a single component of a BER open type extension by decoding an open type value and appending it to the list of open type objects.

**Parameters:**

- buffer* Decode message buffer object

**3.37.2.4 virtual void DecodeEventComponent (Asn1BerDecodeBuffer *buffer*)** [virtual]

This method decodes a single component of a BER open type extension by decoding an open type value and appending it to the list of open type objects, this function also triggers event handler code, with element name "..."

**Parameters:**

- buffer* Decode message buffer object

**3.37.2.5 virtual [Asn1OpenType](#) DecodeOpenType (Asn1PerDecodeBuffer *buffer*, bool *present*, int *index*)**  
[virtual]

This method decodes a single open type extension item in a SEQUENCE or SET construct using the packed encoding rules (PER). It will then add the item to the open extension element list.

**Parameters:**

- buffer* Decode message buffer object
- present* Flag indicating whether element is present
- index* Index of element in the object array

**Returns:**

- Decoded open type

### 3.37.2.6 **override void Encode (Asn1PerOutputStream *outs*)** [virtual]

This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER). Also throws any exception thrown by the underlying Asn1PerOutputStream.

#### **Parameters:**

*outs* PER Output Stream object

#### **Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER) and writes it into the stream.

Also throws any exception thrown by the underlying Asn1BerOutputStream.

#### **Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating element is explicitly tagged

#### **Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

### 3.37.2.8 **override void Encode (Asn1XerEncoder *buffer*)** [virtual]

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

#### **Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### **Parameters:**

*buffer* Encode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

#### **Returns:**

Length of encoded component

Reimplemented from [Asn1Type](#).

### 3.37.2.11 **virtual void EncodeExtBits (Asn1PerEncodeBuffer *buffer*)** [virtual]

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

#### **Parameters:**

*buffer* Encode message buffer object

### 3.37.2.12 **virtual void SetOpenType (Asn1OpenType *obj*, int *index*)** [virtual]

This method will add the given open type object to the open extension element list at the given index.

#### **Parameters:**

*obj* Open type object

*index* Index in open type list where element is to be placed

### 3.37.2.13 **virtual void ShrinkArray (int *numrecs*)** [virtual]

This method adjusts the size of the open type component array downward to the given size value.

#### **Parameters:**

*numrecs* Number of entries the array should hold

### 3.37.2.14 **override System.String ToString ()**

This method will return a string representation of the open extension value. The format is the ASN.1 value format for each open type in the extension.

#### **Returns:**

Stringified representation of the value

### 3.37.3 Member Data Documentation

#### 3.37.3.1 [NonSerialized] System.Collections.ArrayList **mValue**

**Initial value:**

```
new System.Collections.ArrayList ()
```

The value is a list of [Asn1OpenType](#) objects. Each of these objects contains a fully encoded extension item.

## 3.38 Asn1OpenType Class Reference

Inherits [Asn1OctetString](#).

Inherited by [Asn1ChoiceExt](#).

### 3.38.1 Detailed Description

This is a container class for holding the an ASN.1 open type value.

#### Public Member Functions

- [Asn1OpenType](#) ([Asn1EncodeBuffer](#) buffer)
- [Asn1OpenType](#) (byte[] data, int offset, int nbytes)
- [Asn1OpenType](#) (byte[] data)
- [Asn1OpenType](#) ()
- override void [Decode](#) ([Asn1BerDecodeBuffer](#) buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) ([Asn1PerOutputStream](#) outs)
- override void [Encode](#) ([Asn1BerOutputStream](#) outs, bool explicitTagging)
- override void [Encode](#) ([Asn1PerEncodeBuffer](#) buffer)
- override int [Encode](#) ([Asn1BerEncodeBuffer](#) buffer, bool explicitTagging)
- override System.String [ToString](#) ()

#### Protected Attributes

- [NonSerialized] internal [Asn1EncodeBuffer](#) [mEncodeBuffer](#)
- [NonSerialized] internal int [mLength](#)
- [NonSerialized] internal bool [mTextEncoding](#)

### 3.38.2 Constructor & Destructor Documentation

#### 3.38.2.1 [Asn1OpenType](#) ()

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

#### 3.38.2.2 [Asn1OpenType](#) (byte[] data)

This constructor initializes an open type from the given byte array. The array is assumed to contain a previously encoded message component.

#### Parameters:

*data* Byte array containing a previously encoded message component.

### 3.38.2.3 [Asn1OpenType](#) (`byte[] data`, `int offset`, `int nbytes`)

This constructor initializes the open type from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

#### Parameters:

*data* Byte array containing an octet string in binary form.

*offset* The offset in array at which to begin copy.

*nbytes* Number of bytes to copy from target array

### 3.38.2.4 [Asn1OpenType](#) ([Asn1EncodeBuffer](#) *buffer*)

This constructor initializes an open type using an encoded component. This can be used if a header (for example, a ROSE header) is being prepended to a pre-encoded component.

#### Parameters:

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

## 3.38.3 Member Function Documentation

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

This method decodes an ASN.1 open type value.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

### 3.38.3.2 `override void Encode` ([Asn1PerOutputStream](#) *outs*) [virtual]

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

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

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

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

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

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

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

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

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

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

This method encodes an ASN.1 open type value. The value is assumed to be an already-encoded message component and will be copied to the encoded buffer. An optimization is available in which no copy will be performed if the encoded component is already present in the encode buffer. This is done if the form of constructor specifying only a component length is used.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

#### Returns:

Length of encoded component

Reimplemented from [Asn1OctetString](#).

Reimplemented in [Asn1ChoiceExt](#).

### 3.38.3.6 **override System.String ToString ()**

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

#### **Returns:**

Stringified representation of the value

Reimplemented from [Asn1OctetString](#).

## 3.38.4 **Member Data Documentation**

### 3.38.4.1 **[NonSerialized] internal Asn1EncodeBuffer mEncodeBuffer** [protected]

The encode buffer into which component type will be encoded. /p>

### 3.38.4.2 **[NonSerialized] internal int mLength** [protected]

Length of the pre-encoded component. /p>

### 3.38.4.3 **[NonSerialized] internal bool mTextEncoding** [protected]

Use UTF encoding rule for string conversion. /p>

## 3.39 Asn1OutputStream Class Reference

### 3.39.1 Detailed Description

This abstract class implements the base output stream to encode ASN.1 messages.

#### Public Member Functions

- [Asn1OutputStream](#) (System.IO.Stream *os*)
- override void [Close](#) ()
- override void [Flush](#) ()
- override int [Read](#) (byte[] buffer, int offset, int count)
- override long [Seek](#) (long offset, System.IO.SeekOrigin origin)
- override void [SetLength](#) (long value)
- override void [Write](#) (System.Byte[] b, int off, int len)
- virtual void [Write](#) (byte[] b)
- override void [WriteByte](#) (byte b)
- virtual void [WriteByte](#) (int b)

#### Protected Attributes

- internal System.IO.Stream *os*

#### Properties

- override bool [CanRead](#) [get]
- override bool [CanSeek](#) [get]
- override bool [CanWrite](#) [get]
- override long [Length](#) [get]
- override long [Position](#) [get, set]

### 3.39.2 Constructor & Destructor Documentation

#### 3.39.2.1 [Asn1OutputStream](#) (System.IO.Stream *os*)

This constructor creates an output stream object.

##### Parameters:

*os* The underlying System.IO.Stream object.

### 3.39.3 Member Function Documentation

#### 3.39.3.1 override void [Close](#) ()

Closes this output stream and releases any system resources associated with this stream. The general contract of `close` is that it closes the output stream. A closed stream cannot perform output operations and cannot be reopened.

##### Exceptions:

*System.IO.IOException* An error occurred while trying to close the stream.

### 3.39.3.2 override void Flush ()

Flushes this output stream and forces any buffered output bytes to be written out. The general contract of `flush` is that calling it is an indication that, if any bytes previously written have been buffered by the implementation of the output stream, such bytes should immediately be written to their intended destination.

#### Exceptions:

*System.IO.IOException* An I/O error occurs.

*System.ObjectDisposedException* The stream is closed.

### 3.39.3.3 override int Read (byte[] buffer, int offset, int count)

This method always throws `NotSupportedException`. [Asn1OutputStream](#) doesn't support reading.

#### Parameters:

*buffer* When this method returns, contains the specified byte array with the values between `offset` and `(offset + count - 1)` replaced by the bytes read from the current source.

*offset* The byte offset in array at which to begin reading.

*count* The maximum number of bytes to read.

#### Returns:

The total number of bytes read into the buffer.

#### Exceptions:

*System.NotSupportedException* The stream does not support reading.

### 3.39.3.4 override long Seek (long offset, System.IO.SeekOrigin origin)

Sets the current position of this stream to the given value.

#### Parameters:

*offset* The point relative to origin from which to begin seeking.

*origin* Specifies the beginning, the end, or the current position as a reference point for origin, using a value of type `SeekOrigin`.

#### Returns:

The new position in the stream.

#### Exceptions:

*System.IO.IOException* An I/O error occurs.

*System.NotSupportedException* The stream does not support seeking, such as if the `FileStream` is constructed from a pipe or console output.

*System.ArgumentException* Attempted seeking before the beginning of the stream.

*System.ObjectDisposedException* Methods were called after the stream was closed.

### 3.39.3.5 override void SetLength (long value)

Sets the length of this stream to the given value.

#### Parameters:

*value* The new length of the stream.

#### Exceptions:

*System.IO.IOException* An I/O error has occurred.

*System.NotSupportedException* The stream does not support both writing and seeking.

*System.ArgumentOutOfRangeException* Attempted to set the value parameter to less than 0.

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

Writes *len* bytes from the specified byte array starting at offset *off* to this output stream.

#### Parameters:

*b* The byte array data to be written.

*off* The offset in array at which to begin write.

*len* the number of bytes to write.

#### Exceptions:

*System.ArgumentNullException* array is a null reference

*System.ArgumentException* offset and count describe an invalid range in array.

*System.ArgumentOutOfRangeException* offset or count is negative.

*System.IO.IOException* An I/O error occurs.

*System.ObjectDisposedException* The stream is closed.

*System.NotSupportedException* The current stream instance does not support writing.

### 3.39.3.7 virtual void Write (byte[] b) [virtual]

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

#### Parameters:

*b* the data.

#### Exceptions:

*System.ArgumentNullException* array is a null reference

*System.ArgumentException* offset and count describe an invalid range in array.

*System.ArgumentOutOfRangeException* offset or count is negative.

*System.IO.IOException* An I/O error occurs.

*System.ObjectDisposedException* The stream is closed.

*System.NotSupportedException* The current stream instance does not support writing.

### 3.39.3.8 override void WriteByte (byte b)

Writes the specified byte to this output stream. The general contract for `Write` is that one byte is written to the output stream. The byte to be written is the eight low-order bits of the argument `b`. The 24 high-order bits of `b` are ignored.

#### Parameters:

*b* the byte.

#### Exceptions:

*System.ObjectDisposedException* The stream is closed.

*System.NotSupportedException* The stream does not support writing.

### 3.39.3.9 virtual void WriteByte (int b) [virtual]

Writes the specified byte to this output stream. The general contract for `Write` is that one byte is written to the output stream. The byte to be written is the eight low-order bits of the argument `b`. The 24 high-order bits of `b` are ignored.

#### Parameters:

*b* the byte.

#### Exceptions:

*System.ObjectDisposedException* The stream is closed.

*System.NotSupportedException* The stream does not support writing.

## 3.39.4 Member Data Documentation

### 3.39.4.1 internal System.IO.Stream os [protected]

C# Stream object

## 3.39.5 Property Documentation

### 3.39.5.1 override bool CanRead [get]

Gets a value indicating whether the current stream supports reading.

Value: false, the stream doesn't supports reading.

### 3.39.5.2 override bool CanSeek [get]

Gets a value indicating whether the current stream supports seeking.

Value: true if the stream supports seeking; otherwise, false.

### 3.39.5.3 override bool CanWrite [get]

Gets a value indicating whether the current stream supports writing.

Value: true if the stream supports writing; otherwise, false.

#### 3.39.5.4 override long Length [get]

Gets the length in bytes of the stream.

Value: A long value representing the length of the stream in bytes.

#### Exceptions:

*System.NotSupportedException* CanSeek for this stream is false.

*System.IO.IOException* An I/O error occurs, such as the file being closed.

#### 3.39.5.5 override long Position [get, set]

Gets or sets the current position of this stream.

Value: The current position of this stream.

#### Exceptions:

*System.NotSupportedException* The stream does not support seeking.

*System.IO.IOException* An I/O error occurs.

*System.ArgumentOutOfRangeException* Attempted to set the position to a negative value.

*System.IO.EndOfStreamException* Attempted seeking past the end of a stream that does not support this.

## 3.40 Asn1PrintableString Class Reference

Inherits [Asn18BitCharString](#).

### 3.40.1 Detailed Description

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

#### Public Member Functions

- [Asn1PrintableString](#) (System.String data)
- [Asn1PrintableString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.40.2 Constructor & Destructor Documentation

#### 3.40.2.1 [Asn1PrintableString](#) ()

The default constructor creates an empty string object.

#### 3.40.2.2 [Asn1PrintableString](#) (System.String data)

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

#### Parameters:

*data* value string

### 3.40.3 Member Function Documentation

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

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

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

Throws Exception, if any exception thrown by the underlying System.IO.Stream.

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.40.4 Member Data Documentation

### 3.40.4.1 new readonly Asn1Tag \_TAG [static]

The \_TAG constant describes the universal tag for this data type (UNIVERSAL 19).

Reimplemented from [Asn1Type](#).

## 3.41 Asn1Real Class Reference

Inherits [Asn1Type](#).

### 3.41.1 Detailed Description

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

#### Public Member Functions

- [Asn1Real](#) (double value)
- [Asn1Real](#) ()
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- override void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- virtual void [EncodeValue](#) (Asn1XmlEncoder buffer)
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (double value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

#### Public Attributes

- [NonSerialized] double [mValue](#)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.41.2 Constructor & Destructor Documentation

#### 3.41.2.1 [Asn1Real](#) ()

The default constructor sets the double value to zero.

#### 3.41.2.2 [Asn1Real](#) (double *value*)

This constructor creates an REAL object from a double value.

#### Parameters:

*value* double value

### 3.41.3 Member Function Documentation

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

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

**Parameters:**

*buffer* PER Decode message buffer object

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

#### 3.41.3.5 override void Encode (Asn1PerOutputStream *outs*) [virtual]

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

Also throws any exception thrown by the Asn1PerOutputStream.

**Parameters:**

*outs* PER Output Stream object

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

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

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1Type](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* PER Encode message buffer object

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length of component or negative status value

Reimplemented from [Asn1Type](#).

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

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

#### Parameters:

*buffer* Encode message buffer object

*attrName* Attribute name

Reimplemented from [Asn1Type](#).

### 3.41.3.12 virtual void EncodeValue (Asn1XmlEncoder *buffer*) [virtual]

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

#### Parameters:

*buffer* Encode message buffer object

### 3.41.3.13 **override bool Equals (System.Object value)**

Determines whether the specified Object is equal to the current Object.

#### **Parameters:**

*value* The Object to compare with the current Object. Object should be instance of [Asn1Real](#).

#### **Returns:**

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

### 3.41.3.14 **virtual bool Equals (double value) [virtual]**

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

#### **Parameters:**

*value* The double value to compare with the current Object.

#### **Returns:**

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

### 3.41.3.15 **override int GetHashCode ()**

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

#### **Returns:**

A hash code for the current Object.

### 3.41.3.16 **override System.String ToString ()**

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

#### **Returns:**

Stringified representation of the value

## 3.41.4 **Member Data Documentation**

### 3.41.4.1 **new readonly Asn1Tag \_TAG [static]**

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 9).

Reimplemented from [Asn1Type](#).

### 3.41.4.2 **[NonSerialized] double mValue**

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

## 3.42 Asn1RelativeOID Class Reference

Inherits [Asn1ObjectIdentifier](#).

### 3.42.1 Detailed Description

This is a container class for holding the components of an ASN.1 relative object identifier value.

#### Public Member Functions

- [Asn1RelativeOID](#) (int[] value)
- [Asn1RelativeOID](#) ()
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- new static readonly [Asn1Tag\\_TAG](#)

### 3.42.2 Constructor & Destructor Documentation

#### 3.42.2.1 [Asn1RelativeOID](#) ()

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

#### 3.42.2.2 [Asn1RelativeOID](#) (int[] value)

This constructor initializes the object identifier from the given array of integer subidentifier values.

#### Parameters:

*value* Array of subidentifiers

### 3.42.3 Member Function Documentation

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

This method decodes an ASN.1 relative object identifier value using the packed encoding rules (PER).

**Parameters:**

*buffer* Decode message buffer object

Reimplemented from [Asn1ObjectIdentifier](#).

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

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

**Parameters:**

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1ObjectIdentifier](#).

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

Reimplemented from [Asn1ObjectIdentifier](#).

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

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

**Parameters:**

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

Reimplemented from [Asn1ObjectIdentifier](#).

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

This method encodes an ASN.1 relative object identifier value using the packed encoding rules (PER).

The value to be encoded is stored in the `mValue` public member variable within this class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

**Parameters:**

*outs* PER Output Stream object

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1ObjectIdentifier](#).

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

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

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

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1ObjectIdentifier](#).

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

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

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1ObjectIdentifier](#).

**3.42.3.8 override void Encode (Asn1XerEncoder buffer, System.String elemName) [virtual]**

This method encodes an ASN.1 RELATIVE-OID value using the XML encoding rules (XER).

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1ObjectIdentifier](#).

### 3.42.3.9 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method encodes an ASN.1 relative object identifier value using the packed encoding rules (PER). The value to be encoded is stored in the `mValue` public member variable within this class.

#### Parameters:

*buffer* Encode message buffer object

Reimplemented from [Asn1ObjectIdentifier](#).

### 3.42.3.10 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 relative object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length of encoded component in octets

Reimplemented from [Asn1ObjectIdentifier](#).

## 3.42.4 Member Data Documentation

### 3.42.4.1 new static readonly Asn1Tag \_TAG [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 13).

Reimplemented from [Asn1ObjectIdentifier](#).

## 3.43 Asn1SeqOrderException Class Reference

Inherits [Asn1Exception](#).

### 3.43.1 Detailed Description

This class defines the 'ASN.1 sequence order' exception that is thrown when an element is received in a SEQUENCE construct that is not in the correct order..

#### Public Member Functions

- [Asn1SeqOrderException \(\)](#)

### 3.43.2 Constructor & Destructor Documentation

#### 3.43.2.1 [Asn1SeqOrderException \(\)](#)

This constructor creates an exception object with a textual message describing the element that was received out-of-order.

## 3.44 Asn1Status Class Reference

### 3.44.1 Detailed Description

This class defines common constants used in the run-time and generated code. Note that all error reporting in the C# version is done via exceptions. Therefore, there are very few status values defined in the class.

#### Public Attributes

- const int `INDEFLEN` = - 9999

### 3.44.2 Member Data Documentation

#### 3.44.2.1 const int `INDEFLEN` = - 9999

This constant indicates an indefinite length field was parsed

## 3.45 Asn1T61String Class Reference

Inherits [Asn1VarWidthCharString](#).

### 3.45.1 Detailed Description

This is a container class for holding the components of an ASN.1 teletex (T61) string value.

#### Public Member Functions

- [Asn1T61String](#) (System.String data)
- [Asn1T61String](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.45.2 Constructor & Destructor Documentation

#### 3.45.2.1 [Asn1T61String](#) ()

The default constructor creates an empty string object.

#### 3.45.2.2 [Asn1T61String](#) (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

#### Parameters:

*data* String value of T61String

### 3.45.3 Member Function Documentation

#### 3.45.3.1 override void [Decode](#) (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 T61String value including the UNIVERSAL tag value and length if explicit tagging is specified.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

### 3.45.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 T61String value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

### 3.45.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 T61String type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.45.4 Member Data Documentation

### 3.45.4.1 new readonly Asn1Tag \_TAG [static]

The \_TAG constant describes the universal tag for this data type (UNIVERSAL 20).

Reimplemented from [Asn1Type](#).

## 3.46 Asn1Tag Class Reference

### 3.46.1 Detailed Description

This is a container class for holding the components of an ASN.1 tag value.

#### Public Member Functions

- [Asn1Tag](#) (short tagclass, short form, int idCode)
- [Asn1Tag](#) ()
- bool [Equals](#) ([Asn1Tag](#) tag)
- virtual bool [Equals](#) (short tagclass, short form, int idCode)
- virtual bool [IsEOC](#) ()
- override System.String [ToString](#) ()

#### Public Attributes

- const short [APPL](#) = (short) (0x40)
- const short [Bit8Mask](#) = (short) (0x80)
- const short [ClassMask](#) = (short) (0xC0)
- const short [CONS](#) = (short) (0x20)
- const short [CTXT](#) = (short) (0x80)
- const bool [EXPL](#) = true
- const short [EXTIDCODE](#) = (short) (0x1F)
- const short [FormMask](#) = (short) (0x20)
- const short [IDMask](#) = (short) (0x1F)
- const bool [IMPL](#) = false
- const short [L7BitsMask](#) = (short) (0x7f)
- [NonSerialized] short [mClass](#)
- [NonSerialized] short [mForm](#)
- [NonSerialized] int [mIDCode](#)
- const short [PRIM](#) = (short) (0x00)
- const short [PRIV](#) = (short) (0xC0)
- const short [UNIV](#) = (short) (0x00)

#### Static Public Attributes

- static readonly [Asn1Tag](#) [ENUM](#)
- static readonly [Asn1Tag](#) [EOC](#)
- static readonly [Asn1Tag](#) [SEQUENCE](#)
- static readonly [Asn1Tag](#) [SET](#)

#### Properties

- virtual bool [Constructed](#) [get]

## 3.46.2 Constructor & Destructor Documentation

### 3.46.2.1 [Asn1Tag](#) ()

The default constructor initializes all fields to zero

### 3.46.2.2 [Asn1Tag](#) (*short tagclass, short form, int idCode*)

This constructor initializes all fields to the given values

#### Parameters:

*tagclass* Tag class value (UNIV, APPL, CTXT, or PRIV)

*form* Tag form value (PRIM or CONS)

*idCode* Tag identifier code

## 3.46.3 Member Function Documentation

### 3.46.3.1 `bool Equals` ([Asn1Tag](#) *tag*)

This method compares this tag with the given tag value for equality.

#### Parameters:

*tag* [Asn1Tag](#) object to which this tag is to be compared

#### Returns:

`true` if the specified Tags are equal; otherwise, `false`.

### 3.46.3.2 `virtual bool Equals` (*short tagclass, short form, int idCode*) `[virtual]`

This method compares this tag with the given tag value for equality.

#### Parameters:

*tagclass* Tag class value (UNIV, APPL, CTXT, or PRIV)

*form* Tag form value (PRIM or CONS)

*idCode* Tag identifier code

#### Returns:

`true` if the specified Tags are equal; otherwise, `false`.

### 3.46.3.3 `virtual bool IsEOC` () `[virtual]`

This method tests if the tag is an end-of-contents (EOC) tag.

#### Returns:

True if tag is an EOC.

### 3.46.3.4 override System.String ToString ()

This method will return a formatted string representing the tag value. The form is "[&lt;class&gt; &lt;ID&gt;]" (i.e. the ASN.1 standard syntax for a tag value).

#### Returns:

Formatted tag string

## 3.46.4 Member Data Documentation

### 3.46.4.1 const short APPL = (short) (0x40)

Mask value for an APPLICATION tag

### 3.46.4.2 const short Bit8Mask = (short) (0x80)

Bit 8 (MSB) octet mask value

### 3.46.4.3 const short ClassMask = (short) (0xC0)

Mask value to mask the class bits from a tag

### 3.46.4.4 const short CONS = (short) (0x20)

Mask value for CONSTRUCTED form

### 3.46.4.5 const short CTXT = (short) (0x80)

Mask value for a context-specific tag

### 3.46.4.6 readonly Asn1Tag ENUM [static]

ASN.1 ENUMERATED type tag value

### 3.46.4.7 readonly Asn1Tag EOC [static]

ASN.1 end-of-contents (EOC) type tag value

### 3.46.4.8 const bool EXPL = true

This specifies that explicit tagging should be used.

### 3.46.4.9 const short EXTIDCODE = (short) (0x1F)

Mask value for extended tag identifier indicator

**3.46.4.10** `const short FormMask = (short) (0x20)`

Mask value to mask the form bit from a tag

**3.46.4.11** `const short IDMask = (short) (0x1F)`

Mask value to mask the tag ID bits from a tag

**3.46.4.12** `const bool IMPL = false`

This specifies that implicit tagging should be used.

**3.46.4.13** `const short L7BitsMask = (short) (0x7f)`

Lower 7 bits octet mask value

**3.46.4.14** `[NonSerialized] short mClass`

Tag class value (UNIV, APPL, CTXT, or PRIV)

**3.46.4.15** `[NonSerialized] short mForm`

Tag form value (PRIM or CONS)

**3.46.4.16** `[NonSerialized] int mIDCode`

Tag ID code

**3.46.4.17** `const short PRIM = (short) (0x00)`

Mask value for PRIMITIVE form

**3.46.4.18** `const short PRIV = (short) (0xC0)`

Mask value for a PRIVATE tag

**3.46.4.19** `readonly Asn1Tag SEQUENCE [static]`

ASN.1 SEQUENCE type tag value

**3.46.4.20** `readonly Asn1Tag SET [static]`

ASN.1 SET type tag value

**3.46.4.21** `const short UNIV = (short) (0x00)`

Mask value for a UNIVERSAL tag

### **3.46.5 Property Documentation**

#### **3.46.5.1 virtual bool Constructed [get]**

Gets the tag is constructed.

Value: `true` if tag is constructed; otherwise `false`.

## 3.47 Asn1Time Class Reference

Inherits [Asn18BitCharString](#).

Inherited by [Asn1GeneralizedTime](#), and [Asn1UTCTime](#).

### 3.47.1 Detailed Description

This is a base class for holding the components of an ASN.1 generalized and universal times string value.

#### Public Member Functions

- [Asn1Time](#) (System.String data, short typeCode, bool useDerRules)
- [Asn1Time](#) (short typeCode, bool useDerRules)
- virtual void [Clear](#) ()
- virtual int [CompareTo](#) (System.Object other)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- override void [Encode](#) (Asn1PerOutputStream outs)
- virtual void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging, [Asn1Tag](#) tag)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override bool [Equals](#) (System.Object value)
- virtual int [GetDiff](#) ()
- override int [GetHashCode](#) ()
- virtual System.DateTime [GetTime](#) ()
- abstract void [ParseString](#) (System.String data)
- virtual void [ParseXmlString](#) (System.String data)
- virtual void [SetDiff](#) (int inMinutes)
- virtual void [SetDiff](#) (int dhour, int dminute)
- virtual void [SetTime](#) (System.DateTime time)

#### Static Public Member Functions

- static void [PutInteger](#) (System.Text.StringBuilder data, int width, int value)

#### Public Attributes

- const int [Apr](#) = 4
- const int [April](#) = 4
- const int [Aug](#) = 8
- const int [August](#) = 8
- const int [Dec](#) = 12
- const int [December](#) = 12
- const int [Feb](#) = 2
- const int [February](#) = 2
- const int [Jan](#) = 1
- const int [January](#) = 1
- const int [Jul](#) = 7
- const int [July](#) = 7

- const int [Jun](#) = 6
- const int [June](#) = 6
- const int [Mar](#) = 3
- const int [March](#) = 3
- const int [May](#) = 5
- const int [Nov](#) = 11
- const int [November](#) = 11
- const int [Oct](#) = 10
- const int [October](#) = 10
- const int [Sep](#) = 9
- const int [September](#) = 9

## Protected Member Functions

- abstract internal bool [CompileString](#) ()
- internal override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength, [Asn1Tag](#) tag)
- internal override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging, [Asn1Tag](#) tag)
- virtual internal void [Init](#) ()
- virtual internal void [PutInteger](#) (int width, int value)
- virtual internal void [SafeParseString](#) ()

## Static Protected Member Functions

- static internal char [CharAt](#) (System.String s, int index)
- static internal int [ParseInt](#) (System.String str, [IntHolder](#) off, int len)

## Protected Attributes

- [NonSerialized] internal int [day](#)
- [NonSerialized] internal bool [derRules](#)
- [NonSerialized] internal int [diffHour](#)
- [NonSerialized] internal int [diffMin](#)
- [NonSerialized] internal int [hour](#)
- [NonSerialized] internal int [minute](#)
- [NonSerialized] internal int [month](#)
- [NonSerialized] internal bool [parsed](#)
- [NonSerialized] internal string [secFraction](#)
- [NonSerialized] internal int [second](#)
- [NonSerialized] internal bool [utcFlag](#)
- [NonSerialized] internal int [year](#)

## Properties

- virtual int [Day](#) [get, set]
- virtual bool [DER](#) [set]
- virtual int [DiffHour](#) [get, set]
- virtual int [DiffMinute](#) [get]
- virtual string [Fraction](#) [get, set]

- virtual int [Hour](#) [get, set]
- virtual int [Minute](#) [get, set]
- virtual int [Month](#) [get, set]
- virtual int [Second](#) [get, set]
- virtual bool [UTC](#) [get, set]
- virtual int [Year](#) [get, set]

## 3.47.2 Constructor & Destructor Documentation

### 3.47.2.1 [Asn1Time](#) (short *typeCode*, bool *useDerRules*)

This constructor creates an empty time string object.

#### Parameters:

*typeCode* Integer constant from [Asn1Type](#) class ([Asn1Type.GeneralTime](#) or [Asn1Type.UTCtime](#)).  
*useDerRules* 'true' if time string should be encoded with DER/PER.

#### See also:

<seealso cref=Asn1Type.[GeneralTime](#) [GeneralTime](#) code  
 <seealso cref=Asn1Type.[UTCtime](#) [UTCtime](#) code

### 3.47.2.2 [Asn1Time](#) (System.String *data*, short *typeCode*, bool *useDerRules*)

This constructor creates a time string from String value.

#### Parameters:

*data* String representation of time value  
*typeCode* Integer constant from [Asn1Type](#) class ([Asn1Type.GeneralTime](#) or [Asn1Type.UTCtime](#)).  
*useDerRules* 'true' if time string should be encoded with DER/PER.

#### See also:

<seealso cref=Asn1Type.[GeneralTime](#) [GeneralTime](#) code  
 <seealso cref=Asn1Type.[UTCtime](#) [UTCtime](#) code

## 3.47.3 Member Function Documentation

### 3.47.3.1 static internal char [CharAt](#) (System.String *s*, int *index*) [static, protected]

Returns the character at the specified index in the specified string. If index is out of bounds, then '\u0000' character will be returned.

#### Parameters:

*s* String value  
*index* Index of Character in the string

#### Returns:

Character at the specified index. or '\u0000'.

### 3.47.3.2 virtual void Clear () [virtual]

This method clears time string. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

Reimplemented in [Asn1UTCTime](#).

### 3.47.3.3 virtual int CompareTo (System.Object *other*) [virtual]

This method compares this object with [Asn1Time](#) class instance or with System.DateTime instance. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

#### Parameters:

*other* the Object to be compared.

#### Returns:

The difference in Ticks with the specified object.

Reimplemented in [Asn1GeneralizedTime](#), and [Asn1UTCTime](#).

### 3.47.3.4 abstract internal bool CompileString () [protected, pure virtual]

Compiles new time string according X.680 (clauses 41, 42) and ISO 8601.

#### Returns:

true, if succeed, or false code, if error.

Implemented in [Asn1GeneralizedTime](#), and [Asn1UTCTime](#).

### 3.47.3.5 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) Decode method. It throws an exception because it should never be invoked by compiler generated code.

#### Parameters:

*buffer* PER Encode message buffer object

Reimplemented from [Asn18BitCharString](#).

### 3.47.3.6 internal override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*, [Asn1Tag](#) *tag*) [protected, virtual]

This method decodes an ASN.1 Time value including the UNIVERSAL tag value and length if explicit tagging is specified. It is a protected method that can only be accessed by objects subclassed from this type.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

*tag* Universal tag to apply

Reimplemented from [Asn1CharString](#).

### 3.47.3.7 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes ASN.1 Time type, using the XML schema encoding rules(asn2xsd).

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

Reimplemented from [Asn1CharString](#).

### 3.47.3.8 override void Encode (Asn1PerOutputStream *outs*) [virtual]

This method encodes and writes to stream an ASN.1 time string value using the standard Packed Encoding Rules (PER) encode method.

Also throws any exception thrown by the Asn1PerOutputStream.

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn18BitCharString](#).

### 3.47.3.9 virtual void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*, Asn1Tag *tag*) [virtual]

This method encodes and writes to the stream an ASN.1 time string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

*tag* Universal tag to apply

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.3.10 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) encode method. It throws an exception because it should never be invoked by compiler generated code.

#### Parameters:

*buffer* PER Encode message buffer object

Reimplemented from [Asn18BitCharString](#).

### 3.47.3.11 internal override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*, Asn1Tag *tag*) [protected, virtual]

This method encodes ASN.1 time string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified (the universal identifier must be provided by the caller).

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

*tag* Universal tag to apply

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1CharString](#).

### 3.47.3.12 override bool Equals (System.Object *value*)

This method compares this object with [Asn1Time](#) class instance or with Calendar instance. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

#### Parameters:

*value* The Object to compare with the current Object. Object should be instance of [Asn1Time](#) or System.DateTime.

#### Returns:

true if the specified Object is equal to the current Object; otherwise, false.

Reimplemented from [Asn1CharString](#).

### 3.47.3.13 virtual int GetDiff () [virtual]

This method returns the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes. The UTC time is the sum of the local time and positive or negative time difference. Note that the return value may differentiate for different inherited [Asn1Time](#) classes.

#### Returns:

The negative or positive difference, in minutes, between the time zone of the object and UTC time (-12\*60..+12\*60) is returned if the operation is successful.

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.47.3.14 override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

**Returns:**

A hash code for the current Object.

Reimplemented from [Asn1CharString](#).

**3.47.3.15 virtual System.DateTime GetTime () [virtual]**

This method converts the time string to the System.DateTime value. If time represented as UTC time plus or minus difference time, then the result System.DateTime will ignore zone offset, as it doesn't support it. Note that the return value may differentiate for different inherited [Asn1Time](#) classes.

**Returns:**

The System.DateTime value.

**3.47.3.16 virtual internal void Init () [protected, virtual]**

This method initializes the [Asn1Time](#) class member variables.

Reimplemented in [Asn1UTCTime](#).

**3.47.3.17 static internal int ParseInt (System.String str, IntHolder off, int len) [static, protected]**

Parses integer value from String.

**Parameters:**

*str* string is containing integer to be parsed.

*off* The offset in array at which to begin parse.

*len* number of digits to be parsed.

**Returns:**

Parsed integer value.

**3.47.3.18 abstract void ParseString (System.String data) [pure virtual]**

This method parses passed time string. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

**Parameters:**

*data* String representation of Time to be parsed.

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Implemented in [Asn1GeneralizedTime](#), and [Asn1UTCTime](#).

**3.47.3.19 virtual void ParseXmlString (System.String data) [virtual]**

This method parses the given time string value. String must be in XML schema dateTime format. It will throw and exception if the string is not in the valid time format.

**Parameters:**

*data* The time string value to be parsed.

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.47.3.20 virtual internal void PutInteger (int width, int value) [protected, virtual]**

Puts integer in string buffer

**Parameters:**

*width* number of digits

*value* value to be put

**3.47.3.21 static void PutInteger (System.Text.StringBuilder data, int width, int value) [static]**

Puts integer in string buffer

**Parameters:**

*data* destination buffer

*width* number of digits in integer value

*value* integer value to be added

**3.47.3.22 virtual internal void SafeParseString () [protected, virtual]**

This method internally calls the ParseString method for this class. But will not throwing any exception, if error in string format.

### 3.47.3.23 virtual void SetDiff (int *inMinutes*) [virtual]

This method sets the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes. The UTC time is the sum of the local time and positive or negative time difference. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

#### Parameters:

*inMinutes* The negative or positive difference, in minutes, between the time zone of the object and UTC time (-12\*60..+12\*60).

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.3.24 virtual void SetDiff (int *dhour*, int *dminute*) [virtual]

This method sets the hour and minute components of the difference between the time zone of the object and Coordinated Universal Time (UTC). The UTC time is the sum of the local time and positive or negative time difference. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

#### Parameters:

*dhour* The negative or positive hour component of the difference between the time zone of the object and UTC time (-12..+12).

*dminute* The negative or positive minute component of the difference between the time zone of the object and UTC time (-59..+59).

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.3.25 virtual void SetTime (System.DateTime *time*) [virtual]

This method converts the System.DateTime value to time string. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

#### Parameters:

*time* The System.DateTime time value.

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented in [Asn1UTCTime](#).

## 3.47.4 Member Data Documentation

### 3.47.4.1 const int [Apr](#) = 4

Constant to represent April month

**3.47.4.2 const int April = 4**

Constant to represent April month

**3.47.4.3 const int Aug = 8**

Constant to represent August month

**3.47.4.4 const int August = 8**

Constant to represent August month

**3.47.4.5 [NonSerialized] internal int day [protected]**

Day of month component. Legal interval is 1..31.

**3.47.4.6 const int Dec = 12**

Constant to represent December month

**3.47.4.7 const int December = 12**

Constant to represent December month

**3.47.4.8 [NonSerialized] internal bool derRules [protected]**

Indicates DER is used (or CER/PER).

**3.47.4.9 [NonSerialized] internal int diffHour [protected]**

Zone offset's hour component. Legal interval is -12..+12.

**3.47.4.10 [NonSerialized] internal int diffMin [protected]**

Zone offset's minute component. Legal interval is -59..+59.

**3.47.4.11 const int Feb = 2**

Constant to represent February month

**3.47.4.12 const int February = 2**

Constant to represent February month

**3.47.4.13 [NonSerialized] internal int hour [protected]**

Hour component. Legal interval is 0..23.

**3.47.4.14** `const int Jan = 1`

Constant to represent January month

**3.47.4.15** `const int January = 1`

Constant to represent January month

**3.47.4.16** `const int Jul = 7`

Constant to represent July month

**3.47.4.17** `const int July = 7`

Constant to represent July month

**3.47.4.18** `const int Jun = 6`

Constant to represent June month

**3.47.4.19** `const int June = 6`

Constant to represent June month

**3.47.4.20** `const int Mar = 3`

Constant to represent March month

**3.47.4.21** `const int March = 3`

Constant to represent March month

**3.47.4.22** `const int May = 5`

Constant to represent May month

**3.47.4.23** `[NonSerialized] internal int minute [protected]`

Minute component. Legal interval is 0..59.

**3.47.4.24** `[NonSerialized] internal int month [protected]`

Month component. Legal interval is 1..12.

**3.47.4.25** `const int Nov = 11`

Constant to represent November month

**3.47.4.26** `const int November = 11`

Constant to represent November month

**3.47.4.27** `const int Oct = 10`

Constant to represent October month

**3.47.4.28** `const int October = 10`

Constant to represent October month

**3.47.4.29** `[NonSerialized] internal bool parsed` [protected]

Indicates string parsed or not.

**3.47.4.30** `[NonSerialized] internal string secFraction` [protected]

Second's fraction component. Legal interval is 0..INF.

**3.47.4.31** `[NonSerialized] internal int second` [protected]

Second component. Legal interval is 0..59.

**3.47.4.32** `const int Sep = 9`

Constant to represent September month

**3.47.4.33** `const int September = 9`

Constant to represent September month

**3.47.4.34** `[NonSerialized] internal bool utcFlag` [protected]

Indicates UTC flag ('Z') set or not.

**3.47.4.35** `[NonSerialized] internal int year` [protected]

Year component. Legal interval is 0..9999.

## **3.47.5 Property Documentation**

**3.47.5.1** `virtual int Day` [get, set]

Gets or Sets the day of month number component of the time value. The number of the first day in month is 1, the number of the last day may be in interval from 28 to 31. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Day of month component (1..31)

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.47.5.2 virtual bool DER** [set]

Sets the 'use DER' flag which enforces the DER rules when time strings are constructed or parsed.

Value: true for DER rule; otherwise false

**3.47.5.3 virtual int DiffHour** [get, set]

Gets or Sets the hour component of the difference between the time zone of the object and Coordinated Universal Time (UTC). The UTC time is the sum of the local time and positive or negative time difference. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: The negative or positive hour component of the difference between the time zone of the object and UTC time (-12..+12)

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.47.5.4 virtual int DiffMinute** [get]

Gets or Sets the minute component of the difference between the time zone of the object and Coordinated Universal Time (UTC). The UTC time is the sum of the local time and positive or negative time difference. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: The negative or positive minute component of the difference between the time zone of the object and UTC time (-59..+59)

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.47.5.5 virtual string Fraction** [get, set]

Gets or Sets the second's decimal fraction component of the time value. Second's decimal fraction is represented by one digit from 0 to 9. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Second's decimal fraction component (0..9)

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented in [Asn1UTCTime](#).

### 3.47.5.6 virtual int Hour [get, set]

Gets or Sets the hour component of the time value. As the ISO 8601 is based on the 24-hour timekeeping system, two digits represent hours from 00 to 23. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Hour component (0..23)

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.5.7 virtual int Minute [get, set]

Gets or Sets the minute component of the time value. Minutes are represented by two digits from 00 to 59. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Minute component (0..59)

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.5.8 virtual int Month [get, set]

Gets or Sets the month number component of the time value. The number of January is 1, February - 2, December - 12. You may use enumerating values for months decoding: [Asn1Time.January](#), [Asn1Time.February](#), etc. Also you can use short aliases for months: [Asn1Time.Jan](#), [Asn1Time.Feb](#), etc. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Month component (1..12)

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.5.9 virtual int Second [get, set]

Gets or Sets the second component of the time value. Seconds are represented by two digits from 00 to 59. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Second component (0..59)

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.47.5.10 virtual bool UTC [get, set]

Gets or Sets the UTC flag state. If the UTC flag is true, then the time is an UTC time and symbol 'Z' is added at the end of time string. Otherwise, it is a local time.

Value: UTC flag state.

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.47.5.11 virtual int Year** [get, set]

Gets or Sets the year component of the time value. Note that the behaviour value may differentiate for different inherited [Asn1Time](#) classes.

Value: Year component (full 4 digits).

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented in [Asn1UTCTime](#).

## 3.48 Asn1TraceHandler Class Reference

Inherits [Asn1NamedEventHandler](#).

### 3.48.1 Detailed Description

This class is a standard named event handler for printing the data in an encoded message in a human-readable format. Note that this handler will work with data encoded using any of the encoding rules (BER, DER, or PER).

### Public Member Functions

- [Asn1TraceHandler](#) (System.IO.StreamWriter ps)
- [Asn1TraceHandler](#) ()
- virtual void [Characters](#) (System.String svalue, short typeCode)
- virtual void [EndElement](#) (System.String name, int index)
- virtual void [StartElement](#) (System.String name, int index)

### 3.48.2 Constructor & Destructor Documentation

#### 3.48.2.1 [Asn1TraceHandler](#) ()

This constructor sets the output stream to standard output.

#### 3.48.2.2 [Asn1TraceHandler](#) (System.IO.StreamWriter ps)

This constructor sets the output stream to the given StreamWriter.

#### Parameters:

*ps* StreamWriter object

### 3.48.3 Member Function Documentation

#### 3.48.3.1 virtual void [Characters](#) (System.String *svalue*, short *typeCode*) [virtual]

The characters callback method is invoked when content (primitive data) is encountered. A stringified representation of the parsed value is returned.

#### Parameters:

*svalue* Stringified representation of the parsed value. The representation will be in ASN.1 value format.

*typeCode* Identifier specifying the type of the parsed data variable. The enumerated list of values that might appear here is provided in the the [Asn1Type](#) class (see the documentation on this class for a full list of the names).

Implements [Asn1NamedEventHandler](#).

### 3.48.3.2 virtual void EndElement (System.String name, int index) [virtual]

The endElement callback method is invoked when the end of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is detected.

#### Parameters:

*name* Name of the parsed element.

*index* Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implements [Asn1NamedEventHandler](#).

### 3.48.3.3 virtual void StartElement (System.String name, int index) [virtual]

The StartElement callback method is invoked when the start of an element within a constructed type (SEQUENCE, SET, SEQUENCE OF, SET OF, or CHOICE) is encountered.

#### Parameters:

*name* Name of the parsed element.

*index* Index of element in array. Only used for SEQUENCE OF or SET OF elements. Set to -1 for all others.

Implements [Asn1NamedEventHandler](#).

## 3.49 Asn1Type Class Reference

Inherits [Asn1TypeIF](#).

Inherited by [Asn1BigInteger](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Choice](#), [Asn1Enumerated](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1Real](#), and [Asn1UniversalString](#).

### 3.49.1 Detailed Description

This is the base class for all ASN.1 built-in types.

#### Public Member Functions

- virtual void [Decode](#) (System.Object reader, System.IO.Stream byteStream)
- virtual void [Decode](#) (System.Object reader, System.String xmlURI)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1BerDecodeBuffer buffer)
- virtual void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXML](#) (String buffer, String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs)
- virtual void [Encode](#) (Asn1CerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- virtual void [Encode](#) (Asn1XmlEncoder buffer)
- virtual void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1XerEncoder buffer)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer)
- virtual int [Encode](#) (Asn1BerEncodeBuffer buffer)
- virtual int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- virtual void [EncodeAttribute](#) (Asn1XmlEncoder buffer, System.String attrName)
- virtual void [Indent](#) (System.IO.TextWriter outs, int level)
- virtual bool [IsOpenType](#) ()
- virtual bool [MatchTypeName](#) (System.String typeName)
- virtual void [Pdiag](#) (System.String s)
- virtual void [Print](#) (System.String varName)
- virtual void [Print](#) (System.IO.TextWriter outs, System.String varName, int level)
- void [SetKey](#) (byte[] rtkey)
- virtual void [SetOpenType](#) ()

#### Static Public Member Functions

- static System.String [GetTypeNames](#) (short typeCode)

#### Public Attributes

- const short [BIT\\_STRING](#) = 3
- const short [BMPString](#) = 30
- const short [BOOLEAN](#) = 1
- const short [ENUMERATED](#) = 10

- const short [EOC](#) = 0
- const short [EXTERNAL](#) = 8
- const short [GeneralString](#) = 27
- const short [GeneralTime](#) = 24
- const short [GraphicString](#) = 25
- const short [IA5String](#) = 22
- const short [INTEGER](#) = 2
- const short [NULL](#) = 5
- const short [NumericString](#) = 18
- const short [OBJECT\\_IDENTIFIER](#) = 6
- const short [ObjectDescriptor](#) = 7
- const short [OCTET\\_STRING](#) = 4
- const short [OpenType](#) = 99
- const short [PrintableString](#) = 19
- const short [REAL](#) = 9
- const short [RelativeOID](#) = 13
- const short [SEQUENCE](#) = 16
- const short [SET](#) = 17
- const short [T61String](#) = [TeletexString](#)
- const short [TeletexString](#) = 20
- const short [UniversalString](#) = 28
- const short [UTCTime](#) = 23
- const short [UTF8String](#) = 12
- const short [VideotexString](#) = 21
- const short [VisibleString](#) = 26

## Static Public Attributes

- static readonly [Asn1Tag\\_TAG](#) = new [Asn1Tag](#)([Asn1Tag.UNIV](#), [Asn1Tag.PRIM](#), [Asn1Type.EOC](#))

## Protected Member Functions

- virtual internal int [MatchTag](#) ([Asn1BerDecodeBuffer](#) buffer, [Asn1Tag](#) tag)
- virtual internal int [MatchTag](#) ([Asn1BerDecodeBuffer](#) buffer, short tagClass, short tagForm, int tagIDCode)

## Properties

- virtual int [Length](#) [get]

## 3.49.2 Member Function Documentation

### 3.49.2.1 virtual void Decode ([System.Object reader](#), [System.IO.Stream byteStream](#)) [virtual]

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Also throws any IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

#### Parameters:

*reader* XML reader object

*byteStream* Input byte stream object

**Exceptions:**

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

**3.49.2.2 virtual void Decode (System.Object reader, System.String xmlURI) [virtual]**

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Also throws any IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

**Parameters:**

*reader* XML reader object

*xmlURI* URI of a source

**Exceptions:**

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

**3.49.2.3 virtual void Decode (Asn1PerDecodeBuffer buffer) [virtual]**

This method is the base implementation of the standard Packed Encoding Rules (PER) Decode method. It throws an exception because it should never be invoked. Inherited class implements this method in Compiler generated code.

**Parameters:**

*buffer* PER Encode message buffer object

**Exceptions:**

[Asn1Exception](#) if invoked directly

Implements [Asn1TypeIF](#).

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Time](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

**3.49.2.4 virtual void Decode (Asn1BerDecodeBuffer buffer) [virtual]**

This method is used to Decode a message that is encoded in BER or DER format. This version of the method sets tagging to explicit ([Asn1Tag.EXPL](#)) and implicit length to zero.

**Parameters:**

*buffer* Decode message buffer object

### 3.49.2.5 virtual void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method is used to Decode a message that is encoded in BER or DER format.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating explicit tag should be parsed from the encoded type.

*implicitLength* Length of the contents field (only required if explicit is false).

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

### 3.49.2.6 virtual void DecodeXML (String *buffer*, String *attrs*) [virtual]

This method decodes the XML content of a simple type.

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

### 3.49.2.7 virtual void Encode (Asn1PerOutputStream *outs*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) encode method using output stream. It throws an exception because it should never be invoked by compiler generated code.

Also throws any exception thrown by the [Asn1PerOutputStream](#).

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Time](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

### 3.49.2.8 virtual void Encode (Asn1CerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method writes to the stream an encoded ASN.1 type value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Cer Encoding Rules (CER).

Throws, Exception thrown by [C# System.IO.Stream](#) for I/O error

**Parameters:**

*outs* CER Output Stream object  
*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

**3.49.2.9 virtual void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]**

This method writes to the stream an encoded ASN.1 type value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

**Parameters:**

*outs* BER Output Stream object  
*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

**3.49.2.10 virtual void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix) [virtual]**

This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd). It throws an exception because it should never be invoked. Inherited class implements this method in Compiler generated code.

Also throws any exception thrown by the underlying stream.

**Parameters:**

*buffer* XML Encode message buffer object  
*elemName* XML element name of item  
*nsPrefix* Element namespace value

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1GeneralizedTime](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1UTCTime](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

### 3.49.2.11 virtual void Encode (Asn1XmlEncoder *buffer*) [virtual]

This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd). This method invokes the generated method with element name and attribute name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

#### Parameters:

*buffer* XML Encode message buffer object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

### 3.49.2.12 virtual void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method is the base implementation of the standard XML Encoding Rules (XER) encode method. It throws an exception because it should never be invoked by compiler generated code.

Also throws any exception thrown by the underlying stream.

#### Parameters:

*buffer* XER Encode message buffer object

*elemName* XML element name of item

#### Exceptions:

[Asn1Exception](#) if invoked directly

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1GeneralizedTime](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1UTCTime](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

### 3.49.2.13 virtual void Encode (Asn1XerEncoder *buffer*) [virtual]

This method is the base implementation of the standard XML Encoding Rules (XER) encode method. This method invokes the generated method with element name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

#### Parameters:

*buffer* XER Encode message buffer object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1OpenExt](#).

### 3.49.2.14 virtual void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]

This method is the base implementation of the standard Packed Encoding Rules (PER) encode method. It throws an exception because it should never be invoked. Inherited class implements this method in Compiler generated code.

#### Parameters:

*buffer* PER Encode message buffer object

#### Exceptions:

[Asn1Exception](#) if invoked directly

Implements [Asn1TypeIF](#).

Reimplemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Time](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

### 3.49.2.15 virtual int Encode (Asn1BerEncodeBuffer *buffer*) [virtual]

This method is used to encode a message in BER or DER format. This version of the method sets tagging to explicit ([Asn1Tag.EXPL](#)).

#### Parameters:

*buffer* Decode message buffer object

#### Returns:

Length of component or negative status value

### 3.49.2.16 virtual int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method is used to encode this data type in BER or DER format.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tag should be added to the encoded type.

#### Returns:

Length of component or negative status value

Implements [Asn1TypeIF](#).

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

### 3.49.2.17 virtual void EncodeAttribute (AsnXmlEncoder *buffer*, System.String *attrName*) [virtual]

This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd). It throws an exception because it should never be invoked by compiler generated code.

#### Parameters:

*buffer* XML Encode message buffer object

*attrName* XML attribute name of item

*attribute* Element attribute value

<throws> IOException Any exception thrown by the underlying stream. </throws> <throws> [Asn1Exception](#) Thrown, if operation is failed. </throws>

Reimplemented in [Asn1Boolean](#), [Asn1Integer](#), [Asn1OctetString](#), and [Asn1Real](#).

### 3.49.2.18 static System.String GetTypeName (short *typeCode*) [static]

This method will convert a type code into a type name as defined in the X.680 standard..

#### Parameters:

*typeCode* Type code to be converted

#### Returns:

ASN.1 type name

### 3.49.2.19 virtual void Indent (System.IO.TextWriter *outs*, int *level*) [virtual]

This method will indent three spaces in the given print stream. It is used by the print methods to provide a formatted output of an encoded element value.

#### Parameters:

*outs* Print stream

*level* Indentation level (no of spaces is 3 x this number)

### 3.49.2.20 virtual bool IsOpenType () [virtual]

Returns open type mode for XML encoding/decoding.

#### Returns:

true if open type mode is on.

Implements [Asn1TypeIF](#).

**3.49.2.21** **virtual internal int MatchTag (Asn1BerDecodeBuffer *buffer*, Asn1Tag *tag*)** [protected, virtual]

This method will compare the next parsed tag with the given tag value. If they do not match, an exception will be thrown.

**Parameters:**

*buffer* Decode message buffer object

*tag* Tag value to compare

**Returns:**

Decoded length value

**Exceptions:**

*Asn1TagMatchFailedException* Tag is not equal to expected value

**3.49.2.22** **virtual internal int MatchTag (Asn1BerDecodeBuffer *buffer*, short *tagClass*, short *tagForm*, int *tagIDCode*)** [protected, virtual]

This method will compare the next parsed tag with the given tag value. If they do not match, an exception will be thrown.

**Parameters:**

*buffer* Decode message buffer object

*tagClass* Tag class value (UNIV, APPL, CTXT, or PRIV)

*tagForm* Tag form value (PRIM or CONS)

*tagIDCode* Tag identifier code

**Returns:**

Decoded length value

**Exceptions:**

*Asn1TagMatchFailedException* Tag is not equal to expected value

**3.49.2.23** **virtual bool MatchTypeName (System.String *typeName*)** [virtual]

This method is used to check the outer level tag in an XER message to verify it matches the expected value. This method is overridden by generated code. The default implementation always returns true.

**Parameters:**

*typeName* Type name to compare.

**Returns:**

True if name matches internal name.

### 3.49.2.24 virtual void Pdiag (System.String s) [virtual]

This is a diagnostics print method. It is a shorthand way to invoke the [Diag](#) object's println method.

#### Parameters:

*s* diagnostics message to be printed.

### 3.49.2.25 virtual void Print (System.String varName) [virtual]

This method will format and output a primitive value to the standard console output.

#### Parameters:

*varName* Name of variable

### 3.49.2.26 virtual void Print (System.IO.TextWriter outs, System.String varName, int level) [virtual]

This method will format and output a primitive value to the given print stream.

#### Parameters:

*outs* Print output stream

*varName* Name of variable

*level* Indentation level

Implements [Asn1TypeIF](#).

### 3.49.2.27 void SetKey (byte[] rtkey)

This method is used with the limited run-time to set a run-time key value generated by the compiler to allow the run-time to operate on the licensed hosts. This is not used in the unlimited redistribution versions.

#### Parameters:

*rtkey* Run-time key generated by ASN1C

### 3.49.2.28 virtual void SetOpenType () [virtual]

Sets open type mode for XML encoding/decoding.

Implements [Asn1TypeIF](#).

## 3.49.3 Member Data Documentation

### 3.49.3.1 readonly Asn1Tag \_TAG = new Asn1Tag(Asn1Tag.UNIV, Asn1Tag.PRIM, Asn1Type.EOC) [static]

Will hold tag for possible definitions

Reimplemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

### **3.49.3.2 const short BIT\_STRING = 3**

BIT\_STRING type code = 3

### **3.49.3.3 const short BMPString = 30**

BMPString type code = 30

### **3.49.3.4 const short BOOLEAN = 1**

BOOLEAN type code = 1

### **3.49.3.5 const short ENUMERATED = 10**

ENUMERATED type code = 10

### **3.49.3.6 const short EOC = 0**

EOC type code = 0

### **3.49.3.7 const short EXTERNAL = 8**

EXTERNAL type code = 8

### **3.49.3.8 const short GeneralString = 27**

GeneralString type code = 27

### **3.49.3.9 const short GeneralTime = 24**

GeneralTime type code = 24

### **3.49.3.10 const short GraphicString = 25**

GraphicString type code = 25

### **3.49.3.11 const short IA5String = 22**

IA5String type code = 22

**3.49.3.12 const short INTEGER = 2**

INTEGER type code = 2

**3.49.3.13 const short NULL = 5**

NULL type code = 5

**3.49.3.14 const short NumericString = 18**

NumericString type code = 18

**3.49.3.15 const short OBJECT\_IDENTIFIER = 6**

OBJECT\_IDENTIFIER type code = 6

**3.49.3.16 const short ObjectDescriptor = 7**

ObjectDescriptor type code = 7

**3.49.3.17 const short OCTET\_STRING = 4**

OCTET\_STRING type code = 4

**3.49.3.18 const short OpenType = 99**

OpenType type code = 99

**3.49.3.19 const short PrintableString = 19**

PrintableString type code = 19

**3.49.3.20 const short REAL = 9**

REAL type code = 9

**3.49.3.21 const short RelativeOID = 13**

RELATIVE\_OID type code = 13

**3.49.3.22 const short SEQUENCE = 16**

SEQUENCE type code = 16

**3.49.3.23 const short SET = 17**

SET type code = 17

#### 3.49.3.24 **const short T61String = TeletexString**

T61String type code = TeletexString

#### 3.49.3.25 **const short TeletexString = 20**

TeletexString type code = 20

#### 3.49.3.26 **const short UniversalString = 28**

UniversalString type code = 28

#### 3.49.3.27 **const short UTCTime = 23**

UTCTime type code = 23

#### 3.49.3.28 **const short UTF8String = 12**

UTF8String type code = 12

#### 3.49.3.29 **const short VideotexString = 21**

VideotexString type code = 21

#### 3.49.3.30 **const short VisibleString = 26**

VisibleString type code = 26

### 3.49.4 Property Documentation

#### 3.49.4.1 **virtual int Length** [get]

Gets the length of types that can be bound by a size constraint (BIT STRING, OCTET STRING, character string, and SEQUENCE OF/SET OF). An attempt to invoke it on any other type will cause an exception to be thrown.

**Value:** Length of item in units (for example, bits for BIT STRING, octets for OCTET STRING, etc.)

#### **Exceptions:**

***Asn1InvalidLengthException*** if called by type rather than size constrained (BIT STRING, OCTET STRING, character string, or SEQUENCE OF/SET OF)

Reimplemented in [Asn1BitString](#), [Asn1CharString](#), [Asn1OctetString](#), and [Asn1UniversalString](#).

## 3.50 Asn1TypeIF Interface Reference

Inherited by [Asn1Type](#).

### 3.50.1 Detailed Description

This is the base interface for all ASN.1 built-in types.

#### Public Member Functions

- void [Decode](#) (System.Object reader, System.IO.Stream byteStream)
- void [Decode](#) (System.Object reader, System.String xmlURI)
- void [Decode](#) (Asn1PerDecodeBuffer buffer)
- void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- void [Encode](#) (Asn1PerOutputStream outs)
- void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- void [Encode](#) (Asn1XmlEncoder buffer)
- void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- void [Encode](#) (Asn1XerEncoder buffer)
- void [Encode](#) (Asn1PerEncodeBuffer buffer)
- int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- bool [IsOpenType](#) ()
- void [Print](#) (System.IO.TextWriter outs, System.String varName, int level)
- void [SetOpenType](#) ()

### 3.50.2 Member Function Documentation

#### 3.50.2.1 void Decode (System.Object reader, System.IO.Stream byteStream)

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Throws an IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

#### Parameters:

*reader* XML reader object

*byteStream* Input byte stream object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implemented in [Asn1Type](#).

### 3.50.2.2 void Decode (System.Object *reader*, System.String *xmlURI*)

This method declaration is the signature of the standard XML Encoding Rules (XER) Decode method.

Throws an IO exception from the parser, possibly from a byte stream or character stream supplied by the application.

#### Parameters:

*reader* XML reader object

*xmlURI* URI of a source

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Implemented in [Asn1Type](#).

### 3.50.2.3 void Decode (Asn1PerDecodeBuffer *buffer*)

This method declaration is the signature of the standard Packed Encoding Rules (PER) Decode method.

#### Parameters:

*buffer* PER Encode message buffer object

Implemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Time](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

### 3.50.2.4 void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*)

This method declaration is the signature of the standard Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) Decode method.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating explicit tag should be parsed from the encoded type.

*implicitLength* Length of the contents field (only required if explicit is false).

Implemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Type](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

### 3.50.2.5 void Encode (Asn1PerOutputStream *outs*)

This method declaration is the signature of the streaming oriented PER encode method.

Also throws any exception thrown by the [Asn1PerOutputStream](#).

**Parameters:**

*outs* PER Output Stream object

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Implemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Time](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

**3.50.2.6 void Encode (Asn1BerOutputStream outs, bool explicitTagging)**

This method declaration is the signature of the streaming oriented BER encode method.

Throws, Exception thrown by C# System.IO.Stream for I/O error

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Implemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Type](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

**3.50.2.7 void Encode (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)**

This method declaration is the signature of the standard XML Encoding as specified in the XML schema standard(asn2xsd).

Also throws any exception thrown by the underlying stream.

**Parameters:**

*buffer* XML Encode message buffer object

*elemName* XML element name of item

*nsPrefix* Element namespace value

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Implemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1GeneralizedTime](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UTCTime](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

### 3.50.2.8 void Encode (Asn1XmlEncoder *buffer*)

This method declaration is the signature of the standard XML Encoding as specified in the XML schema standard(asn2xsd). This method invokes the generated method with element name and attribute name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

#### Parameters:

*buffer* XML Encode message buffer object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implemented in [Asn1Type](#).

### 3.50.2.9 void Encode (Asn1XerEncoder *buffer*, System.String *elemName*)

This method declaration is the signature of the standard XML Encoding Rules (XER) encode method.

Also throws any exception thrown by the underlying stream.

#### Parameters:

*buffer* XER Encode message buffer object

*elemName* XML element name of item

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implemented in [Asn1BitString](#), [Asn1Boolean](#), [Asn1CharString](#), [Asn1Enumerated](#), [Asn1GeneralizedTime](#), [Asn1Integer](#), [Asn1Null](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Type](#), [Asn1UTCTime](#), [Asn1UniversalString](#), and [Asn1BigInteger](#).

### 3.50.2.10 void Encode (Asn1XerEncoder *buffer*)

This method declaration is the signature of the standard XML Encoding Rules (XER) encode method. This method invokes the generated method with element name set to null. This will cause the ASN.1 type name to be used as the top-level element name.

Also throws any exception thrown by the underlying stream.

#### Parameters:

*buffer* XER Encode message buffer object

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implemented in [Asn1OpenExt](#), and [Asn1Type](#).

### 3.50.2.11 void Encode (Asn1PerEncodeBuffer *buffer*)

This method declaration is the signature of the standard Packed Encoding Rules (PER) encode method.

#### Parameters:

*buffer* PER Encode message buffer object

Implemented in [Asn18BitCharString](#), [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1Time](#), [Asn1Type](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VarWidthCharString](#), and [Asn1BigInteger](#).

### 3.50.2.12 int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*)

This method declaration is the signature of the standard Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) encode method.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tag should be added to the encoded type.

#### Returns:

Length of component or negative status value

Implemented in [Asn1BMPString](#), [Asn1BitString](#), [Asn1Boolean](#), [Asn1ChoiceExt](#), [Asn1Enumerated](#), [Asn1GeneralString](#), [Asn1GeneralizedTime](#), [Asn1GraphicString](#), [Asn1IA5String](#), [Asn1Integer](#), [Asn1Null](#), [Asn1NumericString](#), [Asn1ObjectDescriptor](#), [Asn1ObjectIdentifier](#), [Asn1OctetString](#), [Asn1OpenExt](#), [Asn1OpenType](#), [Asn1PrintableString](#), [Asn1Real](#), [Asn1RelativeOID](#), [Asn1T61String](#), [Asn1Type](#), [Asn1UTCTime](#), [Asn1UTF8String](#), [Asn1UniversalString](#), [Asn1VideotexString](#), [Asn1VisibleString](#), and [Asn1BigInteger](#).

### 3.50.2.13 bool IsOpenType ()

Returns open type mode for XML encoding/decoding.

#### Returns:

true if open type mode is on.

Implemented in [Asn1Type](#).

### 3.50.2.14 void Print (System.IO.TextWriter *outs*, System.String *varName*, int *level*)

This method declaration is the signature of the standard print method used to print the contents of the object representing the ASN.1 type.

#### Parameters:

*outs* Output print stream

*varName* Name of the variable being printed

*level* Indentation level

Implemented in [Asn1Type](#).

### **3.50.2.15 void SetOpenType ()**

Sets open type mode for XML encoding/decoding.

Implemented in [Asn1Type](#).

## 3.51 Asn1UniversalString Class Reference

Inherits [Asn1Type](#).

### 3.51.1 Detailed Description

This is a container class for holding the components of an ASN.1 Universal string value.

#### Public Member Functions

- [Asn1UniversalString](#) (System.String value)
- [Asn1UniversalString](#) (int[] value)
- [Asn1UniversalString](#) ()
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [DecodeXER](#) (System.String buffer, System.String attrs)
- override void [DecodeXML](#) (System.String buffer, System.String attrs)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerOutputStream outs, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, [Asn1CharSet](#) charSet)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- virtual void [EncodeData](#) (Asn1XmlXerEncoder buffer)
- override bool [Equals](#) (System.Object value)
- override int [GetHashCode](#) ()
- override System.String [ToString](#) ()

#### Public Attributes

- const int [BITSPERCHAR](#) = 32
- [NonSerialized] int[] [mValue](#)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

## Protected Member Functions

- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int nchars, int abpc, int ubpc, [Asn1CharSet](#) charSet, int startIdx)
- virtual internal void [Decode](#) (Asn1PerDecodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerOutputStream outs, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Encode](#) (Asn1PerOutputStream outs, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerOutputStream outs, int nchars, int offset, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet, long lower, long upper)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int abpc, int ubpc, [Asn1CharSet](#) charSet)
- virtual internal void [Encode](#) (Asn1PerEncodeBuffer buffer, int nchars, int offset, int abpc, int ubpc, [Asn1CharSet](#) charSet)

## Protected Attributes

- [NonSerialized] internal System.Text.StringBuilder [mStringBuffer](#)

## Properties

- override int [Length](#) [get]

## 3.51.2 Constructor & Destructor Documentation

### 3.51.2.1 [Asn1UniversalString](#) ()

This constructor creates an empty string that can be used in a Decode method call to receive a string value.

### 3.51.2.2 [Asn1UniversalString](#) (int[] *value*)

This constructor initializes the universal string from the given an array of 32-bit integer value.

#### Parameters:

*value* universal string value as array of 32-bit int

### 3.51.2.3 [Asn1UniversalString](#) (System.String *value*)

This constructor converts a standard C# string value into a universal string.

#### Parameters:

*value* universal string value as string

### 3.51.3 Member Function Documentation

#### 3.51.3.1 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]

This overloaded version of the Decode method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable.

##### Parameters:

- buffer* Decode message buffer object
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing permitted alphabet constraint character set (optional)
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

#### 3.51.3.2 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *nchars*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, int *startIdx*) [protected, virtual]

This method decodes the contents of a UniversalString. This version of the method assumes a permitted alphabet constraint is in place.

##### Parameters:

- buffer* Decode message buffer object
- nchars* Number of characters
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)
- startIdx* Start index to fill in value array

#### 3.51.3.3 virtual internal void Decode (Asn1PerDecodeBuffer *buffer*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [protected, virtual]

This method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes that a permitted alphabet constraint has been specified that would reduce the the number of bits-per-character from the default character set. It also assumes a general length determinant is present (i.e. there is not size constraint). The decoded result is stored in the public `mValue` member variable.

##### Parameters:

- buffer* Decode message buffer object
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)

**3.51.3.4 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*)**  
[virtual]

This overloaded version of the Decode method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The decoded result is stored in the public `mValue` member variable.

**Parameters:**

- buffer* Decode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

**3.51.3.5 virtual void Decode (Asn1PerDecodeBuffer *buffer*, Asn1CharSet *charSet*)** [virtual]

This method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but allows a permitted alphabet character set to be specified. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable.

**Parameters:**

- buffer* Decode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)

**3.51.3.6 override void Decode (Asn1PerDecodeBuffer *buffer*)** [virtual]

This method decodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable.

**Parameters:**

- buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

**3.51.3.7 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*)**  
[virtual]

This method decodes an ASN.1 universal string value including the UNIVERSAL tag value and length if explicit tagging is specified.

**Parameters:**

- buffer* Decode message buffer object
- explicitTagging* Flag indicating element is explicitly tagged
- implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

### 3.51.3.8 virtual void DecodeXER (System.String *buffer*, System.String *attrs*) [virtual]

This method decodes an ASN.1 Universal String value using the XML encoding rules (XER).

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

### 3.51.3.9 override void DecodeXML (System.String *buffer*, System.String *attrs*)

This method decodes an ASN.1 Universal String value using the XML schema encoding rules(asn2xsd).

#### Parameters:

*buffer* String containing data to be decoded

*attrs* Attributes string from element tag

### 3.51.3.10 virtual internal void Encode (Asn1PerOutputStream *outs*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*, long *lower*, long *upper*) [protected, virtual]

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

#### Parameters:

*outs* PER Output Stream object

*abpc* Number of bits per character (aligned)

*ubpc* Number of bits per character (unaligned)

*charSet* Object representing the permitted alphabet constraint character set (optional)

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

#### Exceptions:

[\*Asn1Exception\*](#) Thrown, if operation is failed.

### 3.51.3.11 virtual internal void Encode (Asn1PerOutputStream *outs*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*) [protected, virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

**Parameters:**

- outs* PER Output Stream object
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)

**Exceptions:**

- Asn1Exception* Thrown, if operation is failed.

**3.51.3.12 virtual internal void Encode (Asn1PerOutputStream outs, int nchars, int offset, int abpc, int ubpc, Asn1CharSet charSet) [protected, virtual]**

This method encodes the contents of a UniversalString type. This version assumes a permitted alphabet constraint was specified.

Also throws any exception thrown by the Asn1PerOutputStream.

**Parameters:**

- outs* PER Output Stream object
- nchars* Number of characters from string to encode
- offset* Offset to first char in string to encode
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing permitted alphabet constraint character set (optional)

**Exceptions:**

- Asn1Exception* Thrown, if operation is failed.

**3.51.3.13 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet, long lower, long upper) [virtual]**

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the Asn1PerOutputStream.

**Parameters:**

- outs* PER Output Stream object
- charSet* Object representing the permitted alphabet constraint character set
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

**Exceptions:**

- Asn1Exception* Thrown, if operation is failed.

### 3.51.3.14 virtual void Encode (Asn1PerOutputStream outs, Asn1CharSet charSet) [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

#### Parameters:

*outs* PER Output Stream object

*charSet* Object representing permitted alphabet constraint character set (optional)

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

### 3.51.3.15 override void Encode (Asn1PerOutputStream outs) [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

Also throws any exception thrown by the `Asn1PerOutputStream`.

#### Parameters:

*outs* PER Output Stream object

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from `Asn1Type`.

### 3.51.3.16 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 universal string including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by `C# System.IO.Stream` for I/O error

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from `Asn1Type`.

**3.51.3.17** **override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*)**  
[virtual]

This method encodes an ASN.1 Universal String value with element and namespace prefix name tag using the XML Encoding as specified in the XML schema standard(asn2xsd).

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

*nsPrefix* Element namespace value

Reimplemented from [Asn1Type](#).

**3.51.3.18** **override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*)** [virtual]

This method encodes an ASN.1 Universal String value using the XML encoding rules (XER).

**Parameters:**

*buffer* Encode message buffer object

*elemName* Element name

Reimplemented from [Asn1Type](#).

**3.51.3.19** **virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *abpc*, int *ubpc*, [Asn1CharSet](#) *charSet*, long *lower*, long *upper*)** [protected, virtual]

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present but no permitted alphabet constraint.

The value to encode is stored in the public `mValue` member variable.

**Parameters:**

*buffer* Encode message buffer object

*abpc* Number of bits per character (aligned)

*ubpc* Number of bits per character (unaligned)

*charSet* Object representing the permitted alphabet constraint character set (optional)

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

**3.51.3.20** **virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *abpc*, int *ubpc*, [Asn1CharSet](#) *charSet*)** [protected, virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

**Parameters:**

- buffer* Encode message buffer object
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing the permitted alphabet constraint character set (optional)

**3.51.3.21 virtual internal void Encode (Asn1PerEncodeBuffer *buffer*, int *nchars*, int *offset*, int *abpc*, int *ubpc*, Asn1CharSet *charSet*)** [protected, virtual]

This method encodes the contents of a UniversalString type. This version assumes a permitted alphabet constraint was specified.

**Parameters:**

- buffer* Encode message buffer object
- nchars* Number of characters from string to encode
- offset* Offset to first char in string to encode
- abpc* Number of bits per character (aligned)
- ubpc* Number of bits per character (unaligned)
- charSet* Object representing permitted alphabet constraint character set (optional)

**3.51.3.22 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*, long *lower*, long *upper*)** [virtual]

This overloaded version of the encode method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes both a size constraint and permitted alphabet constraint are present.

The value to encode is stored in the public `mValue` member variable.

**Parameters:**

- buffer* Encode message buffer object
- charSet* Object representing the permitted alphabet constraint character set
- lower* Effective size constraint lower bound
- upper* Effective size constraint upper bound

**3.51.3.23 virtual void Encode (Asn1PerEncodeBuffer *buffer*, Asn1CharSet *charSet*)** [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no size constraints but does allow a permitted alphabet character set to be specified.

The value to encode is stored in the public `mValue` member variable.

**Parameters:**

- buffer* Encode message buffer object
- charSet* Object representing permitted alphabet constraint character set (optional)

### 3.51.3.24 **override void Encode (Asn1PerEncodeBuffer *buffer*)** [virtual]

This method encodes an ASN.1 UniversalString value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable.

#### **Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

### 3.51.3.25 **override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*)** [virtual]

This method encodes an ASN.1 universal string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

#### **Parameters:**

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### **Returns:**

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

### 3.51.3.26 **virtual void EncodeData (Asn1XmlXerEncoder *buffer*)** [virtual]

This method encodes an ASN.1 Universal String value using the XML Encoding as specified in the XML schema standard(asn2xsd).

#### **Parameters:**

*buffer* Encode message buffer object

### 3.51.3.27 **override bool Equals (System.Object *value*)**

This method compares this character string value to the given value for equality.

#### **Parameters:**

*value* The Object to compare with the current Object. Object should be instance of [Asn1UniversalString](#).

#### **Returns:**

`true` if the specified Object is equal to the current Object; otherwise, `false`.

### 3.51.3.28 **override int GetHashCode ()**

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

#### **Returns:**

A hash code for the current Object.

### 3.51.3.29 **override System.String ToString ()**

This method will return a string representation of the value. The format is the ASN.1 value format for this type..

#### **Returns:**

Stringified representation of the value

## 3.51.4 **Member Data Documentation**

### 3.51.4.1 **new readonly Asn1Tag \_TAG** [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 28).

Reimplemented from [Asn1Type](#).

### 3.51.4.2 **const int BITSPERCHAR = 32**

The `BITSPERCHAR` constant specifies the number of bits per character for PER (aligned or unaligned).

### 3.51.4.3 **[NonSerialized] internal System.Text.StringBuilder mStringBuffer** [protected]

Variable holds the string representation of the value

### 3.51.4.4 **[NonSerialized] int [] mValue**

The `mValue` public member variable is used to hold the string value to be encoded or the results of a Decode operation. For `UniversalString`, the characters are stored in an array of 32-bit integer values.

## 3.51.5 **Property Documentation**

### 3.51.5.1 **override int Length** [get]

Gets the length of the character string in characters.

Value: number of characters.

Reimplemented from [Asn1Type](#).

## 3.52 Asn1UTCTime Class Reference

Inherits [Asn1Time](#).

### 3.52.1 Detailed Description

This is a container class for holding the components of an ASN.1 UTC time string value.

#### Public Member Functions

- [Asn1UTCTime](#) (System.String data, bool useDerRules)
- [Asn1UTCTime](#) (System.String data)
- [Asn1UTCTime](#) (bool useDerRules)
- [Asn1UTCTime](#) ()
- override void [Clear](#) ()
- override System.Int32 [CompareTo](#) (System.Object obj)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1XmlEncoder buffer, System.String elemName, System.String nsPrefix)
- override void [Encode](#) (Asn1XerEncoder buffer, System.String elemName)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)
- virtual void [EncodeXMLData](#) (Asn1XmlXerEncoder buffer)
- override void [ParseString](#) (System.String data)
- override void [SetTime](#) (System.DateTime time)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

#### Protected Member Functions

- internal override bool [CompileString](#) ()
- internal override void [Init](#) ()

#### Properties

- override string [Fraction](#) [get, set]
- override int [Year](#) [get, set]

### 3.52.2 Constructor & Destructor Documentation

#### 3.52.2.1 [Asn1UTCTime](#) ()

The default constructor creates an empty time string object.

### 3.52.2.2 [Asn1UTCTime](#) (bool *useDerRules*)

This constructor creates an empty UTCTime string object and allows DER encoding rules to be specified.

#### Parameters:

*useDerRules* 'true' if time string should be encoded with DER/PER.

### 3.52.2.3 [Asn1UTCTime](#) (System.String *data*)

This version of the constructor can be used to set the string `mValue` member variable to the given time string.

#### Parameters:

*data* UTCTime as string

### 3.52.2.4 [Asn1UTCTime](#) (System.String *data*, bool *useDerRules*)

This version of the constructor can be used to set the string `mValue` member variable to the given time string and specify DER encoding rules be used to construct the string.

#### Parameters:

*data* UTCTime as string

*useDerRules* 'true' if time string should be encoded with DER/PER.

## 3.52.3 Member Function Documentation

### 3.52.3.1 `override void Clear ()` [virtual]

Clears out time string.

Reimplemented from [Asn1Time](#).

### 3.52.3.2 `override System.Int32 CompareTo (System.Object obj)` [virtual]

This method compares this object with [Asn1Time](#) class instance or with `System.DateTime` instance. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. Note that the action of this method may differentiate for different inherited [Asn1Time](#) classes.

#### Parameters:

*obj* the Object to be compared.

#### Returns:

The difference in Ticks with the specified object.

Reimplemented from [Asn1Time](#).

### 3.52.3.3 internal override bool CompileString () [protected, virtual]

Compiles new time string according X.680 (clause 42) and ISO 8601.

#### Returns:

true if successful; otherwise false.

Implements [Asn1Time](#).

### 3.52.3.4 override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 UTCTime value including the UNIVERSAL tag value and length if explicit tagging is specified.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

### 3.52.3.5 override void Encode (Asn1XmlEncoder *buffer*, System.String *elemName*, System.String *nsPrefix*) [virtual]

This method encodes ASN.1 UTC Time string into xsd:dateTime format with element and namespace prefix name tag according to the XML Encoding as specified in the w3c XML standard(asn2xsd).

#### Parameters:

*buffer* Encode message buffer object

*elemName* XML element name used to wrap string

*nsPrefix* Element namespace value

Reimplemented from [Asn1CharString](#).

### 3.52.3.6 override void Encode (Asn1XerEncoder *buffer*, System.String *elemName*) [virtual]

This method encodes ASN.1 UTC Time string into xsd:dateTime format with element tag according to the XML Encoding as specified in the itu-t XER standard.

#### Parameters:

*buffer* Encode message buffer object

*elemName* XML element name used to wrap string

Reimplemented from [Asn1CharString](#).

### 3.52.3.7 override void Encode (Asn1BerOutputStream outs, bool explicitTagging) [virtual]

This method encodes and writes to the stream an ASN.1 UTC time string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

### 3.52.3.8 override int Encode (Asn1BerEncodeBuffer buffer, bool explicitTagging) [virtual]

This method encodes an ASN.1 UTCTime type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

### 3.52.3.9 virtual void EncodeXMLData (Asn1XmlXerEncoder buffer) [virtual]

This method encodes ASN.1 UTC Time string into xsd:dateTime format XML dateTime format YYYY-MM-DDTHH:MM:SS[.SSSS]([Z|(+|-)HH:MM]) Asn1 UTCTime format YYYYMMDDHH[MM]([Z|(+|-)HH(MM))

#### Parameters:

*buffer* Encode message buffer object

### 3.52.3.10 internal override void Init () [protected, virtual]

This method initializes the [Asn1UTCTime](#) class member variables.

Reimplemented from [Asn1Time](#).

### 3.52.3.11 override void ParseString (System.String *data*) [virtual]

This method parses passed UTCTime string.

#### Parameters:

*data* The UTCTime string value to be parsed.

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Implements [Asn1Time](#).

### 3.52.3.12 override void SetTime (System.DateTime *time*) [virtual]

This method converts the System.DateTime value to UTCTime string.

#### Parameters:

*time* The System.DateTime value.

#### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Time](#).

## 3.52.4 Member Data Documentation

### 3.52.4.1 new readonly [Asn1Tag\\_TAG](#) [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 23).

Reimplemented from [Asn1Type](#).

## 3.52.5 Property Documentation

### 3.52.5.1 override string [Fraction](#) [get, set]

Gets the Fraction component of the UTC Time. Which is always Zero(i.e. empty string)

Value: Zero or empty string

#### Exceptions:

[Asn1Exception](#) Always thrown for Sets call.

Reimplemented from [Asn1Time](#).

### 3.52.5.2 **override int Year** [get, set]

Gets or Sets the year component of the time value. You may pass 'year' parameter either as two last digits of the year (00 - 99) or as full 4 digits (0 - 9999). Note Gets returns year in full 4 digits format.

Value: Year component (full 4 digits).

#### **Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Time](#).

## 3.53 Asn1UTF8String Class Reference

Inherits [Asn1CharString](#).

### 3.53.1 Detailed Description

This is a container class for holding the components of an ASN.1 UTF-8 string value.

#### Public Member Functions

- [Asn1UTF8String](#) (System.String data)
- [Asn1UTF8String](#) ()
- virtual void [Decode](#) (Asn1PerDecodeBuffer buffer, long lower, long upper)
- override void [Decode](#) (Asn1PerDecodeBuffer buffer)
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- virtual void [Encode](#) (Asn1PerOutputStream outs, long lower, long upper)
- override void [Encode](#) (Asn1PerOutputStream outs)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- virtual void [Encode](#) (Asn1PerEncodeBuffer buffer, long lower, long upper)
- override void [Encode](#) (Asn1PerEncodeBuffer buffer)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.53.2 Constructor & Destructor Documentation

#### 3.53.2.1 [Asn1UTF8String](#) ()

The default constructor creates an empty time string object.

#### 3.53.2.2 [Asn1UTF8String](#) (System.String data)

This constructor can be used to set the UTF8 String `mValue` member variable to the given string value.

##### Parameters:

*data* UTF8 String value

### 3.53.3 Member Function Documentation

#### 3.53.3.1 virtual void [Decode](#) (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This method decodes a sized ASN.1 UTF-8 string value using the packed encoding rules (PER).

##### Parameters:

*buffer* Decode message buffer object

*lower* Lower bound (inclusive) of size constraint

*upper* Upper bound (inclusive) of size constraint

### 3.53.3.2 **override void Decode (Asn1PerDecodeBuffer *buffer*)** [virtual]

This method decodes an ASN.1 UTF-8 string value using the packed encoding rules (PER). The string is assumed to not contain a size constraint.

#### **Parameters:**

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

### 3.53.3.3 **override void Decode (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*)** [virtual]

This method decodes an ASN.1 UTF-8 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This string type uses variable length character encodings.

#### **Parameters:**

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

### 3.53.3.4 **virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, long *upper*)** [virtual]

This method encodes a size-constrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Also throws any exception thrown by the Asn1PerOutputStream.

#### **Parameters:**

*outs* PER Output Stream object

*lower* Lower bound (inclusive) of size constraint

*upper* Upper bound (inclusive) of size constraint

#### **Exceptions:**

[Asn1Exception](#) Thrown, if operation is failed.

### 3.53.3.5 **override void Encode (Asn1PerOutputStream *outs*)** [virtual]

This method encodes an unconstrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

Also throws any exception thrown by the Asn1PerOutputStream.

**Parameters:**

*outs* PER Output Stream object

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

**3.53.3.6 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]**

This method encodes and writes to the stream an ASN.1 UTF8 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

**3.53.3.7 virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*) [virtual]**

This method encodes a size-constrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

**Parameters:**

*buffer* Encode message buffer object

*lower* Lower bound (inclusive) of size constraint

*upper* Upper bound (inclusive) of size constraint

**3.53.3.8 override void Encode (Asn1PerEncodeBuffer *buffer*) [virtual]**

This method encodes an unconstrained ASN.1 UTF-8 string value using the packed encoding rules (PER). The value to be encoded is stored in the 'mValue' public member variable within this class.

**Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

### 3.53.3.9 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 UTF8 string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.53.4 Member Data Documentation

### 3.53.4.1 new readonly [Asn1Tag\\_TAG](#) [static]

The `_TAG` constant describes the universal tag for this data type (UNIVERSAL 12).

Reimplemented from [Asn1Type](#).

## 3.54 Asn1Util Class Reference

### 3.54.1 Detailed Description

This class contains some general purpose static utility functions.

#### Static Public Member Functions

- static System.String [BCDToString](#) (byte[] bcd)
- static byte[] [DecodeBase64Array](#) (byte[] srcArray)
- static byte[] [EncodeBase64Array](#) (byte[] srcArray)
- static byte[] [GetAddressBytes](#) (string ipaddress)
- static int [GetBytesCount](#) (long val)
- static int [GetUlongBytesCount](#) (long val)
- static byte[] [StringToBCD](#) (System.String str)
- static byte[] [StringToTBCD](#) (System.String str)
- static System.String [TBCDToString](#) (byte[] bcd)
- static void [ToArray](#) (System.Collections.ICollection c, System.Object[] objects)
- static byte[] [ToByteArray](#) (System.String sourceString)
- static char[] [ToCharArray](#) (byte[] byteArray)
- static System.String [ToHexString](#) (byte[] b, int offset, int nbytes)
- static System.String [ToHexString](#) (byte b)
- static long [URShift](#) (long number, long bits)
- static long [URShift](#) (long number, int bits)
- static int [URShift](#) (int number, long bits)
- static int [URShift](#) (int number, int bits)
- static void [WriteStackTrace](#) (System.Exception throwable, System.IO.TextWriter stream)

### 3.54.2 Member Function Documentation

#### 3.54.2.1 static System.String [BCDToString](#) (byte[] *bcd*) [static]

Translates a BCD string to an ASCII string. The 0xF half-byte is assumed as end of string indicator.

##### Parameters:

*bcd* the source BCD string

##### Returns:

the ASCII string.

#### 3.54.2.2 static byte [] [DecodeBase64Array](#) (byte[] *srcArray*) [static]

Translates the specified Base64 array into byte array. The resulting array could be converted to the String by new String (byte[]) constructor.

##### Parameters:

*srcArray* Base64 byte array to be translated

**Returns:**

decoded byte array

**3.54.2.3 static byte [] EncodeBase64Array (byte[] *srcArray*) [static]**

Translates the specified byte array to Base64 byte array. The resulting array could be converted to the String by new String (byte[]) constructor.

**Parameters:**

*srcArray* byte array to be translated

**Returns:**

Base64 encoded byte array

**3.54.2.4 static byte [] GetAddressBytes (string *ipaddress*) [static]**

Converts an IPAddress to byte array

**Parameters:**

*ipaddress* String representation of IP Address

**Returns:**

The byte array(size 4) representation of IP address

**3.54.2.5 static int GetBytesCount (long *val*) [static]**

Calculate the number of bytes necessary to represent a signed long value.

**Parameters:**

*val* signed long value.

**Returns:**

the number of bytes.

**3.54.2.6 static int GetUlongBytesCount (long *val*) [static]**

Calculate the number of bytes necessary to represent an unsigned long value.

**Parameters:**

*val* unsigned long value.

**Returns:**

the number of bytes.

### 3.54.2.7 static byte [] StringToBCD (System.String *str*) [static]

Translates an ASCII string to a BCD string. The ASCII string must contain only characters in the range [0..9] & ([A..F] | [a..f]). If the length of the source string is not even, the unused part of the last byte will be set to 0xF.

#### Parameters:

*str* the source ASCII string

#### Returns:

the BCD string as a byte array.

#### Exceptions:

[AsnIValueParseException](#) If invalid characters are in the source string.

### 3.54.2.8 static byte [] StringToTBCD (System.String *str*) [static]

Translates an ASCII string to a TBCD string. The ASCII string must contain only characters in the range [0..9] & ([A..F] | [a..f]). If the length of the source string is not even, the unused part of the last byte will be set to 0xF. TBCD strings differ from BCD strings in that the least significant nibble is set in the upper four bits; so the string "12345" would be translated "0x2143f5".

#### Parameters:

*str* the source ASCII string

#### Returns:

the TBCD string as a byte array.

#### Exceptions:

[AsnIValueParseException](#) If invalid characters are in the source string.

### 3.54.2.9 static System.String TBCDToString (byte[] *bcd*) [static]

Translates a TBCD string to an ASCII string. The 0xF half-byte is assumed as end of string indicator.

#### Parameters:

*bcd* the source TBCD string

#### Returns:

the ASCII string.

### 3.54.2.10 static void ToArray (System.Collections.ICollection *c*, System.Object[] *objects*) [static]

Obtains an array containing all the elements of the collection.

**Parameters:**

*c* The Collection instance, which contains the elements.

*objects* The array into which the elements of the collection will be stored.

**Returns:**

The array containing all the elements of the collection.

**3.54.2.11 static byte [] ToByteArray (System.String *sourceString*) [static]**

Converts a string to an array of bytes

**Parameters:**

*sourceString* The string to be converted

**Returns:**

The new array of bytes

**3.54.2.12 static char [] ToCharArray (byte[] *byteArray*) [static]**

Converts an array of bytes to an array of chars

**Parameters:**

*byteArray* The array of bytes to convert

**Returns:**

The new array of chars

**3.54.2.13 static System.String ToHexString (byte[] *b*, int *offset*, int *nbytes*) [static]**

Convert a array of bytes into a hex string.

**Parameters:**

*b* byte array to be converted to hex string

*offset* start position in byte array

*nbytes* no. of bytes to be converted

**Returns:**

Hex String value

### 3.54.2.14 `static System.String ToHexString (byte b)` [static]

Convert a byte value to a hex string. Unlike the C# built-in function, this will:

- a. not sign extend the byte value out to 32 bits if the MSB is set, and
  - b. put a zero padding byte in front if less than 0xf
- In other words, a character string of length 2 is always returned.

#### Parameters:

*b* byte value

#### Returns:

Hex String value

### 3.54.2.15 `static long URShift (long number, long bits)` [static]

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

#### Parameters:

*number* Number to operate on

*bits* Ammount of bits to shift

#### Returns:

The resulting number from the shift operation

### 3.54.2.16 `static long URShift (long number, int bits)` [static]

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

#### Parameters:

*number* Number to operate on

*bits* Ammount of bits to shift

#### Returns:

The resulting number from the shift operation

### 3.54.2.17 `static int URShift (int number, long bits)` [static]

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

#### Parameters:

*number* Number to operate on

*bits* Ammount of bits to shift

#### Returns:

The resulting number from the shift operation

### 3.54.2.18 `static int URShift (int number, int bits)` [static]

Performs an unsigned bitwise right shift with the specified number. The low-order bits of number are discarded, the remaining bits are shifted right, and the high-order empty bit positions are set to zero.

#### Parameters:

*number* Number to operate on

*bits* Ammount of bits to shift

#### Returns:

The resulting number from the shift operation

### 3.54.2.19 `static void WriteStackTrace (System.Exception throwable, System.IO.TextWriter stream)` [static]

Writes the exception stack trace to the received stream

#### Parameters:

*throwable* Exception to obtain information from

*stream* Output sream used to write to

## 3.55 Asn1Value Class Reference

### 3.55.1 Detailed Description

This class provides methods for parsing and formatting text in ASN.1 value notation.

#### Static Public Member Functions

- static byte[] [ParseString](#) (System.String data)
- static byte[] [ParseString](#) (System.String data, [IntHolder](#) numbits)

### 3.55.2 Member Function Documentation

#### 3.55.2.1 static byte[] [ParseString](#) (System.String data) [static]

This overloaded version of the [ParseString](#) method sets the numbits holder value to null.

##### Parameters:

*data* The ASN.1 value specification text

##### Returns:

byte array value

#### 3.55.2.2 static byte[] [ParseString](#) (System.String data, [IntHolder](#) numbits) [static]

This static method parses the given ASN.1 value text (either a binary or hex data string) and returns the value in a binary byte array. The number of bits is also returned.

Examples of valid value formats are as follows:

Binary string: '11010010111001'B

Hex string: '0fa56920014abc'H

Char string: 'abcdefg'

##### Parameters:

*data* The ASN.1 value specification text

*numbits* Holder to receive number of bits in string

##### Returns:

byte array value

## 3.56 Asn1ValueParseException Class Reference

Inherits [Asn1Exception](#).

### 3.56.1 Detailed Description

This class defines the 'ASN.1 value Parse' exception that is thrown when a string containing an ASN.1 value cannot be parsed.

#### Public Member Functions

- [Asn1ValueParseException](#) (System.String text, int offset)
- [Asn1ValueParseException](#) (System.String text)

### 3.56.2 Constructor & Destructor Documentation

#### 3.56.2.1 [Asn1ValueParseException](#) (System.String text)

This constructor creates an exception object with a textual message describing the expected and parsed tag values.

##### Parameters:

*text* The value string that could not be parsed

#### 3.56.2.2 [Asn1ValueParseException](#) (System.String text, int offset)

This constructor creates an exception object with a textual message describing the expected and parsed tag values. This version allows the offset in the string to also be specified.

##### Parameters:

*text* The value string that could not be parsed

*offset* Offset to error location in string

## 3.57 Asn1VarWidthCharString Class Reference

Inherits [Asn1CharString](#).

Inherited by [Asn1GeneralString](#), [Asn1GraphicString](#), [Asn1ObjectDescriptor](#), [Asn1T61String](#), and [Asn1VideotexString](#).

### 3.57.1 Detailed Description

This is an abstract base class for holding the ASN.1 variable width character string types ([GraphicString](#), [GeneralString](#), [TeletexString](#), [T61String](#), [VideotexString](#), [ObjectDescriptor](#)).

#### Public Member Functions

- virtual void [Decode](#) ([Asn1PerDecodeBuffer](#) buffer, long lower, long upper)
- override void [Decode](#) ([Asn1PerDecodeBuffer](#) buffer)
- virtual void [Encode](#) ([Asn1PerOutputStream](#) outs, long lower, long upper)
- override void [Encode](#) ([Asn1PerOutputStream](#) outs)
- virtual void [Encode](#) ([Asn1PerEncodeBuffer](#) buffer, long lower, long upper)
- override void [Encode](#) ([Asn1PerEncodeBuffer](#) buffer)

#### Public Attributes

- const int [BITSPERCHAR\\_A](#) = 8
- const int [BITSPERCHAR\\_U](#) = 8

#### Protected Member Functions

- internal [Asn1VarWidthCharString](#) ([System.String](#) data, short typeCode)
- internal [Asn1VarWidthCharString](#) (short typeCode)

### 3.57.2 Constructor & Destructor Documentation

#### 3.57.2.1 internal [Asn1VarWidthCharString](#) (short typeCode) [protected]

The default constructor creates an empty string object.

##### Parameters:

*typeCode* Universal ID code for ASN.1 character string

#### 3.57.2.2 internal [Asn1VarWidthCharString](#) ([System.String](#) data, short typeCode) [protected]

This version of the constructor can be used to set the string `mValue` member variable to the given string.

##### Parameters:

*data* Character string

*typeCode* Universal ID code for ASN.1 character string

### 3.57.3 Member Function Documentation

#### 3.57.3.1 virtual void Decode (Asn1PerDecodeBuffer *buffer*, long *lower*, long *upper*) [virtual]

This overloaded version of the Decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present. A permitted alphabet may be passed, but it will be ignored.

The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

##### Parameters:

*buffer* Decode message buffer object

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

#### 3.57.3.2 override void Decode (Asn1PerDecodeBuffer *buffer*) [virtual]

This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints. Decoded characters are assigned as-is to the output string. The decoded result is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

##### Parameters:

*buffer* Decode message buffer object

Reimplemented from [Asn1Type](#).

#### 3.57.3.3 virtual void Encode (Asn1PerOutputStream *outs*, long *lower*, long *upper*) [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value directly into the stream, in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present. A permitted alphabet may be passed, but it will be ignored.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Also throws any exception thrown by the [Asn1PerOutputStream](#).

##### Parameters:

*outs* PER Encode message stream object

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

##### Exceptions:

[Asn1Exception](#) Thrown, if operation is failed.

### 3.57.3.4 **override void Encode (Asn1PerOutputStream *outs*)** [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream. This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

Also throws any exception thrown by the `Asn1PerOutputStream`.

#### **Parameters:**

*outs* PER Encode message stream object

#### **Exceptions:**

[Asn1Exception](#) Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

### 3.57.3.5 **virtual void Encode (Asn1PerEncodeBuffer *buffer*, long *lower*, long *upper*)** [virtual]

This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes a size constraint is present. A permitted alphabet may be passed, but it will be ignored.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

#### **Parameters:**

*buffer* Encode message buffer object

*lower* Effective size constraint lower bound

*upper* Effective size constraint upper bound

### 3.57.3.6 **override void Encode (Asn1PerEncodeBuffer *buffer*)** [virtual]

This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER). This version of the method assumes no permitted alphabet or size constraints.

The value to encode is stored in the public `mValue` member variable in the [Asn1CharString](#) base class.

#### **Parameters:**

*buffer* Encode message buffer object

Reimplemented from [Asn1Type](#).

## 3.57.4 **Member Data Documentation**

### 3.57.4.1 **const int [BITSPERCHAR\\_A](#) = 8**

The `BITSPERCHAR_A` constant specifies the number of bits per character for PER (aligned).

### 3.57.4.2 **const int [BITSPERCHAR\\_U](#) = 8**

The `BITSPERCHAR_U` constant specifies the number of bits per character for PER (unaligned).

## 3.58 Asn1VideotexString Class Reference

Inherits [Asn1VarWidthCharString](#).

### 3.58.1 Detailed Description

This is a container class for holding the components of an ASN.1 videotex string value.

#### Public Member Functions

- [Asn1VideotexString](#) (System.String data)
- [Asn1VideotexString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.58.2 Constructor & Destructor Documentation

#### 3.58.2.1 [Asn1VideotexString](#) ()

The default constructor creates an empty string object.

#### 3.58.2.2 [Asn1VideotexString](#) (System.String data)

This version of the constructor can be used to set the string `mValue` member variable to the given string.

#### Parameters:

*data* string representation of videotex string

### 3.58.3 Member Function Documentation

#### 3.58.3.1 override void [Decode](#) (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 Videotex string value including the UNIVERSAL tag value and length if explicit tagging is specified.

Throws, Exception thrown by C# System.IO.Stream for I/O error

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

**3.58.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*)** [virtual]

This method encodes and writes to the stream an ASN.1 videotex string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

**Parameters:**

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

Reimplemented from [Asn1Type](#).

**3.58.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*)** [virtual]

This method encodes an ASN.1 Videotex String type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

**Parameters:**

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

**Returns:**

Length in octets of encoded component

**Exceptions:**

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

### 3.58.4 Member Data Documentation

**3.58.4.1 new readonly Asn1Tag \_TAG** [static]

The \_TAG constant describes the universal tag for this data type (UNIVERSAL 21).

Reimplemented from [Asn1Type](#).

## 3.59 Asn1VisibleString Class Reference

Inherits [Asn18BitCharString](#).

### 3.59.1 Detailed Description

This is a container class for holding the components of an ASN.1 Visible string value.

#### Public Member Functions

- [Asn1VisibleString](#) (System.String data)
- [Asn1VisibleString](#) ()
- override void [Decode](#) (Asn1BerDecodeBuffer buffer, bool explicitTagging, int implicitLength)
- override void [Encode](#) (Asn1BerOutputStream outs, bool explicitTagging)
- override int [Encode](#) (Asn1BerEncodeBuffer buffer, bool explicitTagging)

#### Static Public Attributes

- static new readonly [Asn1Tag\\_TAG](#)

### 3.59.2 Constructor & Destructor Documentation

#### 3.59.2.1 [Asn1VisibleString](#) ()

The default constructor creates an empty string object.

#### 3.59.2.2 [Asn1VisibleString](#) (System.String data)

This version of the constructor can be used to set the Visible string `mValue` member variable to the given string.

#### Parameters:

*data* string representation of visible string

### 3.59.3 Member Function Documentation

#### 3.59.3.1 override void [Decode](#) (Asn1BerDecodeBuffer *buffer*, bool *explicitTagging*, int *implicitLength*) [virtual]

This method decodes an ASN.1 Visible string value including the UNIVERSAL tag value and length if explicit tagging is specified.

#### Parameters:

*buffer* Decode message buffer object

*explicitTagging* Flag indicating element is explicitly tagged

*implicitLength* Length of contents if implicit

Reimplemented from [Asn1Type](#).

### 3.59.3.2 override void Encode (Asn1BerOutputStream *outs*, bool *explicitTagging*) [virtual]

This method encodes and writes to the stream an ASN.1 visible string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Throws, Exception thrown by C# System.IO.Stream for I/O error

#### Parameters:

*outs* BER Output Stream object

*explicitTagging* Flag indicating explicit tagging should be done

#### Exceptions:

*Asn1Exception* Thrown, if operation is failed.

Reimplemented from [Asn1Type](#).

### 3.59.3.3 override int Encode (Asn1BerEncodeBuffer *buffer*, bool *explicitTagging*) [virtual]

This method encodes an ASN.1 Visible string type. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

#### Parameters:

*buffer* Encode message buffer object

*explicitTagging* Flag indicating explicit tagging should be done

#### Returns:

Length in octets of encoded component

Reimplemented from [Asn1Type](#).

## 3.59.4 Member Data Documentation

### 3.59.4.1 new readonly Asn1Tag \_TAG [static]

The \_TAG constant describes the universal tag for this data type (UNIVERSAL 26).

Reimplemented from [Asn1Type](#).

## 3.60 BigInteger Class Reference

### 3.60.1 Detailed Description

This class represents an ASN.1 INTEGER built-in type. In this case, the values can be greater than 64 bits in size. This class is used in generated source code if the <BigInteger> qualifier is specified in a compiler configuration file.

#### Public Member Functions

- [BigInteger](#) (System.String value, int radix)
- [BigInteger](#) (System.Int64 value)
- [BigInteger](#) (System.String value)
- [BigInteger](#) (byte[] value, int sign)
- [BigInteger](#) ()
- override bool [Equals](#) (System.Object value)
- virtual bool [Equals](#) (long value)
- byte[] [GetData](#) ()
- override int [GetHashCode](#) ()
- void [Init](#) (System.String val, int radix)
- bool [IsNegative](#) ()
- void [SecureDelete](#) ()
- void [SetData](#) (byte[] ivalue)
- override System.String [ToString](#) ()
- System.String [ToString](#) (int radix)

#### Static Public Member Functions

- static implicit operator [BigInteger](#) (long value)

### 3.60.2 Constructor & Destructor Documentation

#### 3.60.2.1 [BigInteger](#) ()

The default constructor sets the big integer value object reference to null.

#### 3.60.2.2 [BigInteger](#) (byte[] value, int sign)

This constructor creates a new big integer object and sets it to the byte[] value passed in.

#### Parameters:

*value* String value

*sign* Can be -1 for negative, 0 for zero, or 1 for positive.

### 3.60.2.3 **BigInteger** (System.String *value*)

This constructor creates a new big integer object and sets it to the string value passed in. String value may contain the prefix that describes the radix: 0x - hexadecimal, 0o - octal, 0b - binary. The string value without prefix assumes decimal value. The optional sign '-' may be specified at the beginning of the string to specify the negative value.

#### Parameters:

*value* String value

### 3.60.2.4 **BigInteger** (System.Int64 *value*)

This constructor creates a new big integer object and sets it to the int value passed in.

#### Parameters:

*value* Integer value

### 3.60.2.5 **BigInteger** (System.String *value*, int *radix*)

This constructor creates a new big integer object and sets it to the string value passed in. String value may contain the prefix that describes the radix: 0x - hexadecimal, 0o - octal, 0b - binary. The string value without prefix assumes decimal value. The optional sign '-' may be specified at the beginning of the string to specify the negative value.

#### Parameters:

*value* String value

*radix* Can be 16 for hexadecimal, 8 for octal, 2 for binary or 10 for decimal

## 3.60.3 Member Function Documentation

### 3.60.3.1 override bool Equals (System.Object *value*)

This method compares this integer value to the given value for equality.

#### Parameters:

*value* The Object to compare with the current Object. Object should be instance of [BigInteger](#).

#### Returns:

true if the specified Object is equal to the current Object; otherwise, false.

### 3.60.3.2 virtual bool Equals (long *value*) [virtual]

This method compares this integer value to the given value for equality.

#### Parameters:

*value* The long value to compare with the current Object.

#### Returns:

true if the specified long value is equal to the current Object; otherwise, false.

### 3.60.3.3 `byte [] GetData ()`

This method provides the byte array representation of the integer value

#### Returns:

byte array for integer value

### 3.60.3.4 `override int GetHashCode ()`

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

#### Returns:

A hash code for the current Object.

### 3.60.3.5 `void Init (System.String val, int radix)`

Translates the String representation of a Integer in the specified radix into a [BigInteger](#). The String representation consists of an optional minus sign followed by a sequence of one or more digits in the specified radix. The String may not contain any extraneous Characters (whitespace, for example).

#### Parameters:

*val* String representation of Integer.

*radix* radix to be used in interpreting string value.

#### Exceptions:

*System.FormatException* *val* is not a valid representation of a [BigInteger](#) in the specified radix, or invalid value of radix.

### 3.60.3.6 `bool IsNegative ()`

This method checks the Integer value is negative.

#### Returns:

true if negative; otherwise false

### 3.60.3.7 `static implicit operator BigInteger (long value) [static]`

Overloaded implicit conversion operator for long to [BigInteger](#) value

### 3.60.3.8 `void SecureDelete ()`

This function clears the current value (by overwriting with zeros). Than sets the value to zero, and passes the old value to garbage collector.

### 3.60.3.9 void SetData (byte[] *ivalue*)

This method sets the Integer value from byte array.

#### Parameters:

*ivalue* byte array of the integer value

#### Returns:

Decoded integer value

#### See also:

<seealso cref=GetData To retrieve Byte Array

### 3.60.3.10 override System.String ToString ()

This method will return a string representation of the integer value. The format is the ASN.1 value format for this type..

#### Returns:

Stringified representation of the value

### 3.60.3.11 System.String ToString (int *radix*)

Returns the String representation of this [BigInteger](#) in the given radix. If the radix is invalid, it will default to 10 (as is the case for `Int32.ToString`). The a minus sign is prepended if appropriate. This method is compatible with `BigInteger(String, int)` constructor.

#### Parameters:

*radix* radix of the String representation.

#### Returns:

String representation of this [BigInteger](#) in the given radix.

#### See also:

<seealso cref=System.Int32.ToString Integer [ToString](#) methods

## 3.61 BooleanHolder Class Reference

### 3.61.1 Detailed Description

A Holder class for a `Boolean` that is used to store "out" and "inout" parameters in methods. If a method has a `boolean` as an "out" or "inout" parameter, the programmer must pass an instance of `BooleanHolder` as the corresponding parameter in the method invocation; for "inout" parameters, the programmer must also fill the "in" value.

**If `myBooleanHolder` is an instance of `BooleanHolder`,**

the value stored in its `value` field can be accessed with `myBooleanHolder.mValue`.

### Public Member Functions

- `BooleanHolder` (bool value)
- `BooleanHolder` ()

### Public Attributes

- bool `mValue`

### 3.61.2 Constructor & Destructor Documentation

#### 3.61.2.1 `BooleanHolder` ()

The default constructor for `BooleanHolder` class

#### 3.61.2.2 `BooleanHolder` (bool value)

This constructor will initialize `BooleanHolder` class with specified boolean value.

**Parameters:**

*value* `Boolean` value

### 3.61.3 Member Data Documentation

#### 3.61.3.1 bool `mValue`

This member variable is where the boolean value is stored.

## 3.62 Diag Class Reference

### 3.62.1 Detailed Description

This class is used for printing diagnostic messages for debugging the run-time components. It allows messages to be easily switched on and off.

#### Public Member Functions

- virtual bool [IsEnabled](#) (int traceLevel)
- virtual bool [IsEnabled](#) ()
- virtual void [Println](#) (System.String s, int traceLevel)
- virtual void [Println](#) (System.String s)
- virtual bool [SetEnabled](#) (bool data)
- virtual int [SetTraceLevel2](#) (int level)

#### Static Public Member Functions

- static void [HexDump](#) (System.IO.Stream istrm, System.IO.StreamWriter ostrm)
- static void [HexDump](#) (byte[] bytes, int traceLevel)
- static void [HexDump](#) (byte[] bytes)
- static [Diag Instance](#) ()
- static void [Prtln](#) (byte[] b, int offset, int nbytes)
- static void [Prtln](#) (byte[] b, int offset, int nbytes, int tl)
- static void [Prtln](#) (System.String s, int traceLevel)
- static void [Prtln](#) (System.String s)
- static int [SetTraceLevel](#) (int level)

#### Properties

- virtual System.IO.StreamWriter [PrintStream](#) [set]

### 3.62.2 Member Function Documentation

#### 3.62.2.1 static void [HexDump](#) (System.IO.Stream *istrm*, System.IO.StreamWriter *ostrm*) [static]

This method prints a formatted hex dump for the contents of the given input stream to the given output stream.

#### Parameters:

*istrm* Input Stream containing data to be dumped

*ostrm* Output Stream to which formatted data is to be written

### 3.62.2.2 `static void HexDump (byte[] bytes, int traceLevel)` [static]

This method prints a formatted hex dump for the contents of the given byte array to the standard output stream, if the given trace level is enabled.

#### Parameters:

*bytes* Byte array containing data to be dumped

*traceLevel* Trace level

### 3.62.2.3 `static void HexDump (byte[] bytes)` [static]

This method prints a formatted hex dump for the contents of the given byte array to the standard output stream.

#### Parameters:

*bytes* Byte array containing data to be dumped

### 3.62.2.4 `static Diag Instance ()` [static]

This method provides the current instance of the [Diag](#) Class

#### Returns:

Current instance of the [Diag](#) class

### 3.62.2.5 `virtual bool IsEnabled (int traceLevel)` [virtual]

This method checks that given trace level message will be printed.

#### Parameters:

*traceLevel* Trace Level

#### Returns:

true if enabled, else false

### 3.62.2.6 `virtual bool IsEnabled ()` [virtual]

This method will enable the diagnostic message printing.

#### Returns:

true if enabled, else false

### 3.62.2.7 `virtual void Println (System.String s, int traceLevel)` [virtual]

This method prints a the diagnostic message, if given trace level is enabled

#### Parameters:

*s* diagnostic message

*traceLevel* Trace Level

### 3.62.2.8 virtual void Println (System.String s) [virtual]

This method prints a the diagnsotic message

#### Parameters:

*s* diagnsotic message

### 3.62.2.9 static void Prtln (byte[] b, int offset, int nbytes) [static]

This method prints a the hex dump of the given byte array to current [Diag](#) class instance

#### Parameters:

*b* byte array containing data

*offset* start offset in the byte array

*nbytes* no of bytes to be printed

### 3.62.2.10 static void Prtln (byte[] b, int offset, int nbytes, int tl) [static]

This method prints a the hex dump of the given byte array to current [Diag](#) class instance, if given trace level is enabled

#### Parameters:

*b* byte array containing data

*offset* start offset in the byte array

*nbytes* no of bytes to be printed

*tl* trace level

### 3.62.2.11 static void Prtln (System.String s, int traceLevel) [static]

This method prints a the diagnsotic message to current [Diag](#) class instance, if given trace level is enabled

#### Parameters:

*s* diagnsotic message

*traceLevel* Trace Level

### 3.62.2.12 static void Prtln (System.String s) [static]

This method prints a the diagnsotic message to current [Diag](#) class instance.

#### Parameters:

*s* diagnsotic message

### 3.62.2.13 **virtual bool SetEnabled (bool *data*)** [virtual]

This method enables or disables the diagnostic message printing.

#### **Parameters:**

*data* true for enabling printing; otherwise false

#### **Returns:**

The stat before setting this stat

### 3.62.2.14 **static int SetTraceLevel (int *level*)** [static]

This method sets the trace level for the current instance of the [Diag Class](#)

#### **Parameters:**

*level* Trace Level

#### **Returns:**

Set trace level

### 3.62.2.15 **virtual int SetTraceLevel2 (int *level*)** [virtual]

This method sets the trace level for this class

#### **Parameters:**

*level* Trace Level

#### **Returns:**

Set trace level

## 3.62.3 **Property Documentation**

### 3.62.3.1 **virtual System.IO.StreamWriter PrintStream** [set]

Sets the System.IO.StreamWriter object to which the diagnostic messages should be written.

Value: Output stream for diagnostic messages

## 3.63 IntHolder Class Reference

### 3.63.1 Detailed Description

A Holder class for an `int` that is used to store "out" and "inout" parameters in methods. If a method has an `int` as an "out" or "inout" parameter, the programmer must pass an instance of `IntHolder` as the corresponding parameter in the method invocation; for "inout" parameters, the programmer must also fill the "in" value.

If `myIntHolder` is an instance of `IntHolder`,

the value stored in its `value` field can be accessed with `myIntHolder.mValue`.

### Public Member Functions

- `IntHolder` (`int` value)
- `IntHolder` ()

### Public Attributes

- `int` `mValue`

### 3.63.2 Constructor & Destructor Documentation

#### 3.63.2.1 `IntHolder` ()

The default constructor for `IntHolder` class

#### 3.63.2.2 `IntHolder` (`int` *value*)

This constructor will initialize `IntHolder` class with specified `int` value.

**Parameters:**

*value* `int` value

### 3.63.3 Member Data Documentation

#### 3.63.3.1 `int` `mValue`

This member variable is where the `int` value is stored.

## 3.64 StringBufferExt Class Reference

### 3.64.1 Detailed Description

This class provides the additional functionality to `StringBuilder` class

#### Static Public Member Functions

- static `StringBuilder Replace` (`StringBuilder sbuf`, `int start`, `int end`, `String str`)

### 3.64.2 Member Function Documentation

#### 3.64.2.1 static `StringBuilder Replace` (`StringBuilder sbuf`, `int start`, `int end`, `String str`) `[static]`

Replaces the characters in a substring of given `StringBuilder` with characters in the specified `String`. The substring begins at the specified `start` and extends to the character at index `end - 1` or to the end of the `StringBuilder` if no such character exists. First the characters in the substring are removed and then the specified `String` is inserted at `start`. (The specified `StringBuilder` will be lengthened to accommodate the specified `String` if necessary.)

#### Parameters:

- sbuf* `StringBuilder` that will have contents.
- start* The beginning index, inclusive.
- end* The ending index, exclusive.
- str* `String` that will replace previous contents.

#### Returns:

- The replaced string builder.

## 3.65 Tokenizer Class Reference

### 3.65.1 Detailed Description

The class performs token processing in strings

#### Public Member Functions

- bool [HasMoreTokens](#) ()
- bool [MoveNext](#) ()
- System.String [NextToken](#) (System.String delimiters)
- System.String [NextToken](#) ()
- string [RemainingString](#) ()
- void [Reset](#) ()
- [Tokenizer](#) (System.String source, System.String delimiters, bool includeDelims)
- [Tokenizer](#) (System.String source, System.String delimiters)
- [Tokenizer](#) (System.String source)

#### Properties

- int [Count](#) [get]
- System.Object [Current](#) [get]

### 3.65.2 Constructor & Destructor Documentation

#### 3.65.2.1 [Tokenizer](#) (System.String *source*)

Initializes a new class instance with a specified string to process

##### Parameters:

*source* String to tokenize

#### 3.65.2.2 [Tokenizer](#) (System.String *source*, System.String *delimiters*)

Initializes a new class instance with a specified string to process and the specified token delimiters to use

##### Parameters:

*source* String to tokenize

*delimiters* String containing the delimiters

#### 3.65.2.3 [Tokenizer](#) (System.String *source*, System.String *delimiters*, bool *includeDelims*)

Initializes a new class instance with a specified string to process, the specified token delimiters to use, and whether the delimiters must be included in the results.

##### Parameters:

*source* String to tokenize

*delimiters* String containing the delimiters

*includeDelims* Determines if delimiters are included in the results.

### 3.65.3 Member Function Documentation

#### 3.65.3.1 bool HasMoreTokens ()

Determines if there are more tokens to return from the source string

**Returns:**

True or false, depending if there are more tokens

#### 3.65.3.2 bool MoveNext ()

Performs the same action as HasMoreTokens.

**Returns:**

True or false, depending if there are more tokens

#### 3.65.3.3 System.String NextToken (System.String *delimiters*)

Returns the next token from the source string, using the provided token delimiters

**Parameters:**

*delimiters* String containing the delimiters to use

**Returns:**

The string value of the token

#### 3.65.3.4 System.String NextToken ()

Returns the next token from the token list

**Returns:**

The string value of the token

#### 3.65.3.5 string RemainingString ()

Returns the rest of the string from current position.

**Returns:**

rest of the string

### **3.65.3.6 void Reset ()**

Does nothing.

## **3.65.4 Property Documentation**

### **3.65.4.1 int Count [get]**

Remaining tokens count

### **3.65.4.2 System.Object Current [get]**

Performs the same action as NextToken.