# Introduction

This document contains details about the design for the new ASN.1 runtime *Asn1Base*.

# Keys of the design

## Reversed Encoding buffer for BER

A BER encoding result is a triple of TLV, where T stands for the Tag of the type, L stands for the Length of the pure encoding result for the data, V stands for the result. It is intuitive to design such an algorithm, which writes Tag, Length, Value to the buffer in order. The question is, Length is determined by Value. To write a Length to the buffer, we have to get Result first. Different Lengths will be encoded into bytes with different sizes. Therefore, we could not write the encoding result for the data to the buffer before Length is writte. We have to store the encoding result in some other place, calculate the Length, write the Length to the buffer, and then copy the result to the buffer. As a consequent, calculating a TLV triple need to do Array.Copy once.

The complex thing is, a structure may be a composition of some other structures. For example, structure A has three fields, of which the type is C, D and E. The encoding result of A is a TLV triple, where T is the Tag, V is the composition of C, D, E’s BER encoding results, and L is the length of V. C, D, E’ BER encoding results are also TLV triples. If C, D, E are also a composition of some other structures, the Value of their BER encoding result are also TLV triples. As we mentioned, calculating a TLV triple need to do Array.Copy once. To calculate A’s encoding result, we will do Array.Copy many times. This is not acceptable.

Currently we use a reversed buffer to improve the performance. A reversed buffer provides two functions:

1. Write a byte to the front of the buffer.
2. Write some bytes to the front of the buffer. The order of the bytes keeps.

With the help of reversed buffer, we could calculate the encoding result for the data first, meanwhile write the result the buffer immediately by function 2, then write the length to the buffer, also by function 2, and finally write the tag by function 1. If the encoding result V is a composition of some other TLV triples, we could calculate them and write them to the buffer by the same way. Note that the fields should be handed reversely as well, i.e., the last field should be handled first, then the last but one…

## Encoding/decoding Framework

The encoding/decoding functionalities are provided by a framework based on OO. Check Asn1Object class for detail.

## Reflection

Attributes are used to specify the ASN.1 part for a structure. Check [User Guide](https://microsoft.sharepoint.com/teams/winteropsh/TestSuiteDevelopment/ASN.1Replacement/ASN.1%20Runtime%20User%20Guide.docx?d=w8f6897fd80ce42a9a9a8973dca61a826) for detail.

# Classes

All the classes belong to namespace *Microsoft.Protocols.TestTools.StackSdk.Asn1*. [Class Diagram](https://microsoft.sharepoint.com/teams/winteropsh/TestSuiteDevelopment/ASN.1Replacement/ASN.1%20Runtime%20Class%20Diagram.vsdx?d=w3117673062e14d9c9c9466c5b7de7716) for the structural classes.

## Asn1Object

Supports all structural classes in the Asn1Base class hierarchy and provides encoding and decoding framework to derived classes. This is the ultimate base class for all structural classes in Asn1Base; it is the root of the hierarchy.

### Inheritance Hierarchy

Asn1Object

*All structures in ASN.1*

### Methods

#### Asn1Tag GetUniversalTag()

Returns the Universal Class Tag of the structure. For each ASN.1 structure, it must have a Universal Class Tag. In the C# class definition, the Universal Class Tag will be specified by attribute *Asn1Tag* with type *Universal*. Check *Asn1Integer* for an example.

#### Asn1Tag GetTopTag()

Returns the top most tag in the buffer while encoding/decoding. According to the encoding/decoding rules of ASN.1, if the structure has an Application Class Tag, then this tag will be returned, otherwise the Universal Class Tag of the structure will be returned. Application Class Tag is used in the same way as Universal Class Tag. Check [ASN.1 Runtime User Guide](https://microsoft.sharepoint.com/teams/winteropsh/TestSuiteDevelopment/ASN.1Replacement/ASN.1%20Runtime%20User%20Guide.docx?d=w8f6897fd80ce42a9a9a8973dca61a826) for detail.

#### bool VerifyConstraints()

Checks whether the stored data meets the constraints based on the ASN.1 definition. Constraints include length constraints and value constraints. This method need to be overrode in the user-defined classes if there are some user-defined constraints for the structure. It will be invoked before the encoding and after the decoding. Check [ASN.1 Runtime User Guide](https://microsoft.sharepoint.com/teams/winteropsh/TestSuiteDevelopment/ASN.1Replacement/ASN.1%20Runtime%20User%20Guide.docx?d=w8f6897fd80ce42a9a9a8973dca61a826) for detail.

#### bool Equals(Object o)

Determines whether the specified object is equal to the current object. This method should be overrode by other classes in the runtime, e.g., Asn1Integer, Asn1HomogenerousComposition and Asn1Choice.

#### string ToString()

Returns a string that represents the current object. This method will be overrode by the string types in the runtime.

#### int LengthBerEncode(IAsn1BerEncodingBuffer buffer, int length)

Encodes a “length” to byte array and stores the result to the given buffer. This method will return the length of the encoding result.

#### int LengthBerDecode(IAsn1DecodingBuffer buffer, out int length)

Decodes a “length” from the buffer. The decoding result will be stored in the second parameter. The method will return the number of the bytes consumed in the buffer to decode the length.

#### int TagBerEncode(IAsn1BerEncodingBuffer buffer, Asn1Tag tag)

Encodes a tag to byte array and stores the result to the given buffer. This method will return the length of the encoding result.

#### int TagBerDecode(IAsn1DecodingBuffer buffer, out Asn1Tag tag)

Decodes a tag from the buffer. The decoding result will be stored in the second parameter. The method will return the number of the bytes that are consumed in the buffer to decode the tag.

#### int ValueBerEncode(IAsn1BerEncodingBuffer buffer)

Encodes the data of the instance to byte array and stores the result to the given buffer. This method will return the length of the encoding result. It is virtual and not implemented in Asn1Object. It should be overrode in the derived classes in the runtime.

#### int ValueBerDecode(IAsn1DecodingBuffer buffer, int length)

Decodes data from the buffer and stores the data in the instance. The method will return the number of the bytes that are consumed in the buffer to decode the data. It is virtual and not implemented in Asn1Object. It should be overrode in the derived classes in the runtime.

#### int BerEncode(IAsn1BerEncodingBuffer buffer, bool explicitTag)

All of the BER encoding for ASN.1 structures are implemented by this method. It gets the TLV (Tag, Length, Value) result by invoking ValueBerEncode, LengthBerEncode and TagBerEncode in order. The result will be written to the given buffer. The length of the result is returned. Note that IAsn1BerEncodingBuffer is reversed, i.e., the latest data written to the buffer appears in the foremost. Therefore, to get the TLV result, the encoding order is Value, Length and Tag. In our test suite, “explicitTag” is always true. The false case is not handled. It is added in case that it will be used in the future.

#### int BerDecode(IAsn1DecodingBuffer buffer)

All of the BER decoding for ASN.1 structures are implemented by this method. It retrieves the data from the buffer by invokeing TagBerDecode, LengthBerDecode and ValueBerDecode in order. The result will be stored in the instance. The number of bytes consumed to decode the data will be returned.

### Remarks

Asn1Object provides the encoding and decoding framework (“BerEncode” and “BerDecode”) for all ASN.1 structures. The framework invokes methods for encoding and decoding Tags, Lengths and Values. The methods for Tags and Lengths are implemented. They should not be overrode unless you would like to handle them in your own way, i.e., in a different way from the Standard (X.690). For the derived structural classes in the runtime, as long as they implement the “ValueBerEncode” and “ValuBerDecode” methods by overriding, the “BerEncode” and “BerDecode” will work. Check *Asn1Integer* for an example.

TODO: Change the two “Get” methods to property.

## Asn1Integer

Represents an INTEGER in ASN.1, which also provides encoding and decoding functionality by the derived methods (“BerEncode” and “BerDecode”) from Asn1Object.

### Inheritance Hierarchy

Asn1Object

Asn1Integer

### Fields and Properties

#### long? Value

Stores data of instances

### Methods

#### byte[] IntegerEncoding(long? Val)

Provides the BER encoding functionality for INTEGER. This method will be invoked by the overrode method “ValueBerEncode”.

#### int GetDeletedCount(byte[] bytesBigEndian, bool negative)

An auxiliary method invoked by “IntegerEncoding”.

#### long IntegerDecoding(byte[] encodingResult)

Provides the BER decoding functionality for INTEGER. This method will be invoked by the overrode method “ValueBerDecode”.

#### byte[] GetExpandedCount(byte[] bytesBigEndianEncodingResult)

An auxiliary method invoked by “IntegerDecoding”.

### Remarks

It can be seen that only “ValueBerEncode” and “ValueBerDecode” are needed to be implemented for Asn1Integer (by invoking “IntegerEncoding” and “IntegerDecoding”). After the implementation and attaching the Universal Class Tag, the class is consistent.

## Asn1Boolean

The design for Asn1Boolean is the same as Asn1Integer.

### Inheritance Hierarchy

Asn1Object

Asn1Boolean

## Asn1String

This is the base class for all string structures (Except for BIT STRING).

### Inheritance Hierarchy

Asn1Object

Asn1String

### Fields and Properties

#### string Value

Stores data of instances.

## Asn1BmpString

Represents a BMP STRING in ASN.1 Definition.

### Inheritance Hierarchy

Asn1Object

Asn1String

Asn1BmpString

### Remarks

Just implement “ValueBerEncode” and “ValueBerDecode” by “Encoding.BigEndianUnicode” and attach the Universal Class Tag.

## Asn1UniversalString

Represents a UniversalString in ASN.1 Definition.

### Inheritance Hierarchy

Asn1Object

Asn1String

Asn1UniversalString

### Remarks

Just implement “ValueBerEncode” and “ValueBerDecode” by “UTF32Encoding” and attach the Universal Class Tag.

## Asn1ByteString

This is the base class of OCTET STRING, GeneralString, NumericString and GenelizedTime.

### Inheritance Hierarchy

Asn1Object

Asn1String

Asn1ByteString

### Fields and Properties

#### string Value

Gets or sets data of instances in string form.

#### byte[] ByteArrayValue

Gets or sets data of instances in byte array form.

### Remarks

Just implement “ValueBerEncode” and “ValueBerDecode” by the transformation between “char” and “int”.

## Asn1OctetString

This is the base class of OCTET STRING, GeneralString, NumericString and GenelizedTime.

### Inheritance Hierarchy

Asn1Object

Asn1String

Asn1ByteString

Asn1OctetString

### Remarks

Just attach a Universal Class Tag.

TODO: The permitted character set need to be added by “VerifyConstraints” method.

## Asn1GeneralString, Asn1NumericString, Asn1GeneralizedTime

### Remarks

Same as Asn1OctetString. Just attach different Universal Class Tags.

TODO: The permitted character set need to be added by “VerifyConstraints” method.

## Asn1BitString

Represents an BIT STRING in ASN.1.

### Inheritance Hierarchy

Asn1Object

Asn1BitString

### Fields and Properties

#### byte[] ByteArrayValue

Stores data of instances.

#### byte unusedBitsInLastByte

Indicates the number of the bits that are not used in the last byte of the data.

#### Int Length

Returns the number of bits in the data.

## Asn1Enumerated

This is the base class of all the ASN.1 ENUMERATED structures.

### Inheritance Hierarchy

Asn1Object

Asn1Integer

Asn1Enumerated

### Fields and Properties

#### List<long?> allowedValues

Stores data of the allowed values. It will be initialized in the constructor by reflection.

#### long? undefinedValue

Represents an undefined value. Used when initializing a new instance without providing any parameters.

## Asn1HomogeneousComposition<T>

This is the base class of SET OF and SEQUENCE OF.

### Inheritance Hierarchy

Asn1Object

Asn1HomogeneousComposition<T>

### Fields and Properties

#### T[] Elements

Stores data in array form.

#### long? undefinedValue

Represents an undefined value. Used when initializing a new instance without providing any parameters.

### Remarks

Since the buffer is reversed in BER encoding, “ValueBerEncode” will encode the data by iterating from the end of “Elements” to the begin.

## Asn1SequenceOf<T>

This is the base class of SEQUENCE OF structure.

### Inheritance Hierarchy

Asn1Object

Asn1HomogeneousComposition<T>

Asn1SequenceOf<T>

### Remarks

Just attach a Universal Class Tag.

## Asn1SetOf<T>

This is the base class of SET OF structure.

### Inheritance Hierarchy

Asn1Object

Asn1HomogeneousComposition<T>

Asn1SetOf<T>

### Remarks

Just attach a Universal Class Tag.

## Asn1HeterogeneousComposition

This is the base class of SET and SEQUENCE.

### Inheritance Hierarchy

Asn1Object

Asn1HeterogeneousComposition

### Methods

#### MemberInfo[] fieldsMemberInfo

Stores the MemberInfo of the fields. It will be initialized by reflection in the constructor.

#### Asn1Tag[] attachedTags

Stores the Context Class Tags for each field. It will be initialized by reflection in the constructor.

#### bool[] fieldOptionalFlags

Stores the flags which specify whether a field is optional. This information will be obtained via the Asn1Field attributes for the fields. If the field corresponding to fieldsMemberInfo[i] is optional, fieldOptionalFlags[i] will be set to true. Otherwise, it will be set to false.

#### void CollectMetadata()

This method will be invoked by constructor and it will initialize fieldsMemberInfo, attachedTags and fieldsOptionalFlags.

#### Asn1Object[] Fields

This property will get all the data in the fields by fieldsMemberInfo.

#### Asn1Object[] GetFieldsTypeInstance()

This property will get new instances of all the fields in the structures. It is used in “BerDecode”.

### Remarks

Since the buffer is reversed in BER encoding, “ValueBerEncode” will encode the data by iterating from the end of “Fields” to the begin.

## Asn1Choice

This is the base class of all CHOICE structures. Fields methods like “fieldsMemberInfo”, “allowedIndices”, “CollectMetadata”, are the same as those in Asn1Enumerated and Asn1HeterogeneousComposition.

### Inheritance Hierarchy

Asn1Object

Asn1Choice

### Fields

#### long? choiceIndexInFieldsMemberInfo

Stores the index of the chosen field in “FieldsMemberInfo”

### Methods

#### void SetData(long? Index, Asn1Object obj)

Stores “obj” into the CHOICE structure. “Index” is the corresponding index of the choice defined in “FieldsMemberInfo”.