ASN1C

ASN.1 Compiler
Version 7.3
Java Runtime
Reference Manual

Objective Systems, Inc. version 7.3 — January 2019

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 ©1997–2019 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.

Package
com.objsys.asn1j.runtime

com.objsys.asn1j.runtime
Class Asn18BitCharString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
```

Direct Known Subclasses:
Asn1VisibleString, Asn1TimeOfDay, Asn1Time, Asn1PrintableString, Asn1NumericString, Asn1IA5String, Asn1Duration, Asn1DateTime, Asn1Date, Asn1AbstractTime

---

public abstract class **Asn18BitCharString**
extends Asn1CharString

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

Field Summary

static final int	BITSPERCHAR_A The BITSPERCHAR_Aconstant specifies the number of bits per character for PER (aligned). Value: **8**
static final int	BITSPERCHAR_U The BITSPERCHAR_Uconstant specifies the number of bits per character for PER (unaligned). Value: **7**

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

protected	Asn18BitCharString(short typeCode) The default constructor creates an empty string object.
protected	Asn18BitCharString(java.lang.String data,short typeCode) This version of the constructor can be used to set the string valuemember variable to the given string.

Method Summary

void	`decode(Asn1OerDecodeBuffer buffer)` Decode the value in accordance with OER.
void	`decode(Asn1OerDecodeBuffer buffer,int length)` Decode the value in accordance with OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,Asn1CharSet charSet)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1OerEncodeBuffer buffer)` Encode the value in accordance with OER.
void	`encode(Asn1OerEncodeBuffer buffer,boolean withLength)` Encode the string, with or without a length determinant.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,Asn1CharSet charSet)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream.
void	`encode(Asn1PerOutputStream out,Asn1CharSet charSet)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream.
void	`encode(Asn1PerOutputStream out,Asn1CharSet charSet,long lower,long upper)` 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).

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

BITSPERCHAR_A

public static final int **BITSPERCHAR_A**

The BITSPERCHAR_Aconstant specifies the number of bits per character for PER (aligned).

BITSPERCHAR_U

public static final int **BITSPERCHAR_U**

The BITSPERCHAR_Uconstant specifies the number of bits per character for PER (unaligned).

Constructors

Asn18BitCharString

protected **Asn18BitCharString**(short typeCode)

The default constructor creates an empty string object.

Parameters:

typeCode - Universal ID code for ASN.1 character string

Asn18BitCharString

protected **Asn18BitCharString**(java.lang.String data,
short typeCode)

This version of the constructor can be used to set the string valuemember variable to the given string.

Parameters:

data - Character string

typeCode - Universal ID code for ASN.1 character string

Methods

(continued on next page)

(continued from last page)

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

Throws:

`java.io.IOException` - for I/O exception

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

`charSet` - Object representing permitted alphabet constraint character set (optional)

Throws:

`java.io.IOException` - for I/O exception

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`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

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

(continued on next page)

(continued from last page)

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

`charSet` - Object representing permitted alphabet constraint character set (optional)

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         Asn1CharSet charSet,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`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

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object

Throws:

(continued on next page)

(continued from last page)

`java.io.IOException -` Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception -` Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out -` PER Encode message stream object
`charSet -` Object representing permitted alphabet constraint character set (optional)

Throws:

`java.io.IOException -` Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception -` Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
         Asn1CharSet charSet,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out -` 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

Throws:

`java.io.IOException -` Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception -` Thrown, if operation is failed.

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode the value in accordance with OER. This class's implementation decodes the string with a length determinant. If a subclass should be decoded without a length determinant, it should override this method to invoke decode(buffer, length). Subclasses may override this to add constraint checks after invoking this method to decode the string.

Parameters:

`buffer -` The buffer to decode from.

Throws:

`java.io.IOException -` for I/O exception

(continued on next page)

(continued from last page)

decode

```
public final void decode(Asn1OerDecodeBuffer buffer,
          int length)
  throws java.io.IOException
```

Decode the value in accordance with OER. This class's implementation decodes a string of the given length. This method is final as I don't see any reason for overriding it.

Parameters:

`buffer` - The buffer to decode from.
`length` - Length of string to decode

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode the value in accordance with OER. This class's implementation invokes encode(buffer, true) to encode the string with a length determinant. If a subclass should be encoded without a length determinant, it should override this to invoke encode(buffer, false).

Parameters:

`buffer` - The buffer to encode to.

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer,
          boolean withLength)
  throws java.io.IOException
```

Encode the string, with or without a length determinant. Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters:

`buffer` - The buffer to encode to.
`withLength` - true if a length determinant should be encoded.

Throws:

`java.io.IOException` - for I/O exception

com.objsys.asn1j.runtime
Class Asn1AbstractTime

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1AbstractTime
```

All Implemented Interfaces:
java.lang.Comparable, java.lang.Cloneable, java.io.Serializable, Asn1TypeIF

Direct Known Subclasses:
Asn1UTCTime, Asn1GeneralizedTime

---

public abstract class **Asn1AbstractTime**
extends Asn18BitCharString
implements Asn1TypeIF, java.io.Serializable, java.lang.Cloneable, java.lang.Comparable

This is a base class for holding the components of an ASN.1 generalized and universal times string value.

Field Summary

static final int	Apr Value: **4**
static final int	April Value: **4**
static final int	Aug Value: **8**
static final int	August Value: **8**
int	day Day of month component.
static final int	Dec Value: **12**
static final int	December Value: **12**
boolean	derRules Indicates DER is used (or CER/PER).

int	diffHour Zone offset's hour component.
int	diffMin Zone offset's minute component.
static final int	Feb Value: **2**
static final int	February Value: **2**
int	hour Hour component.
static final int	Jan Value: **1**
static final int	January Value: **1**
static final int	Jul Value: **7**
static final int	July Value: **7**
static final int	Jun Value: **6**
static final int	June Value: **6**
static final int	Mar Value: **3**
static final int	March Value: **3**
static final int	May Value: **5**
int	minute Minute component.
int	month Month component.
static final int	Nov Value: **11**

static final int	November Value: **11**
static final int	Oct Value: **10**
static final int	October Value: **10**
boolean	parsed Indicates string parsed or not.
java.lang.String	secFraction Second's fraction component.
int	second Second component.
static final int	Sep Value: **9**
static final int	September Value: **9**
boolean	utcFlag Indicates UTC flag ('Z') set or not.
int	year Year component.

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1AbstractTime(short typeCode,boolean useDerRules) This constructor creates an empty time string object.

public	`Asn1AbstractTime(java.lang.String data,short typeCode,boolean useDerRules)` This constructor creates a time string using `data`String.

Method Summary

static char	`charAt(java.lang.String s,int index)` Returns the character at the specified index in the specified string.
void	`clear()` This method clears time string.
int	`compareTo(java.lang.Object other)` This method compares this object with Asn1Time class instance or with Calendar instance.
abstract boolean	`compileString()` Compiles new time string accoring X.680 (clauses 41, 42) and ISO 8601.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength,Asn1Tag tag)` This method decodes an ASN.1 character string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1PerDecodeBuffer buffer)` This method is the base implementation of the standard Packed Encoding Rules (PER) decode method.
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes ASN.1 GeneralizedTime type, using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit,Asn1Tag tag)` This method encodes ASN.1 time string type.
void	`encode(Asn1BerOutputStream out,boolean explicit,Asn1Tag tag)` 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.
void	`encode(Asn1PerEncodeBuffer buffer)` This method is the base implementation of the standard Packed Encoding Rules (PER) encode method.
void	`encode(Asn1PerOutputStream out)` This method encodes and writes to stream an ASN.1 time string value using the standard Packed Encoding Rules (PER) encode method.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes this ASN.1 time into xsd:dateTime format with element and attribute name tag according to the XML Encoding as specified in the XML schema standard(asn2xsd).
void	`encodeXER(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes this ASN.1 time according to XER encoding rules.

void	`encodeXMLData(Asn1XmlXerEncoder buffer)` This method encodes this ASN.1 time string into xsd:dateTime format.
boolean	`equals(java.lang.Object other)` This method compares this object with Asn1Time class instance or with Calendar instance.
abstract boolean	`equals(java.lang.String s)` This method compares this object with a String value.
int	`getDay()` This method returns the day of month number component of the time value.
int	`getDiff()` This method returns the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes.
int	`getDiffHour()` This method returns the hour component of the difference between the time zone of the object and Coordinated Universal Time (UTC).
int	`getDiffMinute()` This method returns the minute component of the difference between the time zone of the object and Coordinated Universal Time (UTC).
java.lang.String	`getFraction()` This method returns the second's decimal fraction component of the time value.
int	`getHour()` This method returns the hour component of the time value.
int	`getMinute()` This method returns the minute component of the time value.
int	`getMonth()` This method returns the month number component of the time value.
int	`getSecond()` This method returns the second component of the time value.
java.util.Calendar	`getTime()` This method converts the time string to the java.util.Calendar value.
boolean	`getUTC()` This method returns the UTC flag state.
int	`getYear()` This method returns the year component of the time value.
void	`init()`
static int	`parseInt(java.lang.String str,IntHolder off,int len)` Parses integer value using String.

abstract void	parseString(java.lang.String string) This method parses passed time string.
void	parseXmlString(java.lang.String string) This method parses the given time string value.
void	putInteger(int width,int value) Puts integer in string buffer
static void	putInteger(java.lang.StringBuffer data,int width,int value) Puts integer in string buffer
void	safeParseString()
void	setDay(int day) This method sets the day of month number component of the time value.
void	setDER(boolean bvalue)
void	setDiff(int inMinutes) This method sets the difference between the time zone of the object and Coordinated Universal Time (UTC), in minutes.
void	setDiff(int dhour,int dminute) This method sets the hour and minute components of the difference between the time zone of the object and Coordinated Universal Time (UTC).
void	setDiffHour(int dhour) This method sets the hour component of the difference between the time zone of the object and Coordinated Universal Time (UTC).
void	setFraction(java.lang.String fraction) This method sets the second's decimal fraction component of the time value.
void	setHour(int hour) This method sets the hour component of the time value.
void	setMinute(int minute) This method sets the minute component of the time value.
void	setMonth(int month) This method sets the month number component of the time value.
void	setSecond(int second) This method sets the second component of the time value.
void	setTime(java.util.Calendar time) This method converts the java.util.Calendar value to time string.
void	setUTC(boolean utc) This method sets the UTC flag state.
void	setYear(int year) This method sets the year component of the time value.

Methods inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

year

protected transient int **year**

Year component. Legal interval is 0..9999.

month

protected transient int **month**

Month component. Legal interval is 1..12.

day

protected transient int **day**

Day of month component. Legal interval is 1..31.

hour

protected transient int **hour**

Hour component. Legal interval is 0..23.

minute

protected transient int **minute**

Minute component. Legal interval is 0..59.

second

`protected transient int` **`second`**

Second component. Legal interval is 0..59.

secFraction

`protected transient java.lang.String` **`secFraction`**

Second's fraction component. Legal interval is 0..INF.

diffHour

`protected transient int` **`diffHour`**

Zone offset's hour component. Legal interval is -12..+12.

diffMin

`protected transient int` **`diffMin`**

Zone offset's minute component. Legal interval is -59..+59.

utcFlag

`protected transient boolean` **`utcFlag`**

Indicates UTC flag ('Z') set or not.

parsed

`protected transient boolean` **`parsed`**

Indicates string parsed or not.

derRules

`protected transient boolean` **`derRules`**

Indicates DER is used (or CER/PER).

January

`public static final int` **`January`**

Jan

`public static final int` **`Jan`**

February

`public static final int` **`February`**

(continued from last page)

Feb

public static final int **Feb**

March

public static final int **March**

Mar

public static final int **Mar**

April

public static final int **April**

Apr

public static final int **Apr**

May

public static final int **May**

June

public static final int **June**

Jun

public static final int **Jun**

July

public static final int **July**

Jul

public static final int **Jul**

August

public static final int **August**

Aug

```
public static final int Aug
```

September

```
public static final int September
```

Sep

```
public static final int Sep
```

October

```
public static final int October
```

Oct

```
public static final int Oct
```

November

```
public static final int November
```

Nov

```
public static final int Nov
```

December

```
public static final int December
```

Dec

```
public static final int Dec
```

Constructors

Asn1AbstractTime

```
public Asn1AbstractTime(short typeCode,
                        boolean useDerRules)
```

This constructor creates an empty time string object.

(continued from last page)

Parameters:

`typeCode` - Integer constant from Asn1Type class (`Asn1Type.GeneralTime`or `Asn1Type.UTCTime`).
`useDerRules` - 'true' if time string should be encoded with DER/PER.

Asn1AbstractTime

```
public Asn1AbstractTime(java.lang.String data,
                        short typeCode,
                        boolean useDerRules)
```

This constructor creates a time string using `data`String.

Parameters:

`typeCode` - Integer constant from Asn1Type class (`Asn1Type.GeneralTime`or `Asn1Type.UTCTime`).
`useDerRules` - 'true' if time string should be encoded with DER/PER.

Methods

init

```
protected void init()
```

---

getYear

```
public int getYear()
  throws Asn1Exception
```

This method returns the year component of the time value. Note that the return value may differentiate for different inherited Asn1Time classes.

Returns:

Year component (full 4 digits) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

getMonth

```
public int getMonth()
  throws Asn1Exception
```

This method returns 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 return value may differentiate for different inherited Asn1Time classes.

Returns:

Month component (1..12) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

getDay

```
public int getDay()
  throws Asn1Exception
```

This method returns 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 return value may differentiate for different inherited Asn1Time classes.

(continued from last page)

Returns:

Day of month component (1..31) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getHour

```
public int getHour()
  throws Asn1Exception
```

This method returns 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 return value may differentiate for different inherited Asn1Time classes.

Returns:

Hour component (0..23) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getMinute

```
public int getMinute()
  throws Asn1Exception
```

This method returns the minute component of the time value. Minutes are represented by two digits from 00 to 59. Note that the return value may differentiate for different inherited Asn1Time classes.

Returns:

Minute component (0..59) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getSecond

```
public int getSecond()
  throws Asn1Exception
```

This method returns the second component of the time value. Seconds are represented by two digits from 00 to 59. Note that the return value may differentiate for different inherited Asn1Time classes.

Returns:

Second component (0..59) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getFraction

```
public java.lang.String getFraction()
  throws Asn1Exception
```

This method returns the second's decimal fraction component of the time value. Second's decimal fraction is represented by from 0 to infinite value. Note that the return value may differentiate for different inherited Asn1Time classes.

Returns:

Second's decimal fraction component is returned if the operation is successful.

Throws:

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

getDiffHour

```
public int getDiffHour()
  throws Asn1Exception
```

This method returns 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 return value may differentiate for different inherited Asn1Time classes.

Returns:

The negative or positive hour component of the difference between the time zone of the object and UTC time (-12..+12) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getDiffMinute

```
public int getDiffMinute()
  throws Asn1Exception
```

This method returns 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 return value may differentiate for different inherited Asn1Time classes.

Returns:

The negative or positive minute component of the difference between the time zone of the object and UTC time (-59..+59) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getDiff

```
public int getDiff()
  throws Asn1Exception
```

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.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getUTC

```
public boolean getUTC()
  throws Asn1Exception
```

This method returns 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.

Returns:

UTC flag state is returned.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

getTime

```
public java.util.Calendar getTime()
  throws Asn1Exception
```

This method converts the time string to the java.util.Calendar value. If time represented as UTC time plus or minus difference time, then the result Calendar will contain ZONE_OFFSET field. Note that the return value may differentiate for different inherited Asn1Time classes. The returned value will be a non-lenient object of Calendar class. It can be changed to lenient object using Calendar.setLenient() function.

Returns:

The java.util.Calendar non-lenient value.

setDER

```
public void setDER(boolean bvalue)
```

setUTC

```
public void setUTC(boolean utc)
  throws Asn1Exception
```

This method 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.

Parameters:

`utc` - UTC flag state.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setYear

```
public void setYear(int year)
  throws Asn1Exception
```

This method sets the year component of the time value. Note that the action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`year` - Year component (full 4 digits).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setMonth

```
public void setMonth(int month)
  throws Asn1Exception
```

This method 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 encoding: Asn1Time.January, Asn1Time.February, etc. Also you can use short aliases for months: Asn1Time.Jan, Asn1Time.Feb, etc. Note that the action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`month` - Month component (1..12).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setDay

```
public void setDay(int day)
  throws Asn1Exception
```

This method 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 action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`day` - Day of month component (1..31).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setHour

```
public void setHour(int hour)
  throws Asn1Exception
```

This method 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 action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`hour` - Hour component (0..23).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setMinute

```
public void setMinute(int minute)
  throws Asn1Exception
```

This method sets the minute component of the time value. Minutes are represented by two digits from 00 to 59. Note that the action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`minute` - Minute component (0..59).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setSecond

```
public void setSecond(int second)
  throws Asn1Exception
```

This method sets the second component of the time value. Seconds are represented by two digits from 00 to 59. Note that the action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`second` - Second component (0..59).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

setFraction

```
public void setFraction(java.lang.String fraction)
  throws Asn1Exception
```

This method 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 action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`fraction` - Second's decimal fraction component (0..9).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

setTime

```
public void setTime(java.util.Calendar time)
  throws Asn1Exception
```

This method converts the java.util.Calendar value to time string. Note that the action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`time` - The calendar time value.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

setDiffHour

```
public void setDiffHour(int dhour)
  throws Asn1Exception
```

This method 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 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) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

setDiff

```
public void setDiff(int dhour,
         int dminute)
  throws Asn1Exception
```

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).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

setDiff

```
public void setDiff(int inMinutes)
  throws Asn1Exception
```

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) is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

clear

```
public void clear()
```

This method clears time string. Note that the action of this method may differentiate for different inherited Asn1Time classes.

---

equals

```
public abstract boolean equals(java.lang.String s)
```

This method compares this object with a String value. Strictly speaking, the contract of equals stipulates identity: a.equals(b) iff b.equals(a). However, it is never the case that a String will equal an Asn1Time object. In this case, then, equals should be taken to mean that the time represented by the string passed in is the same time as the object to which it is compared. This method is implemented in all subclasses of Asn1Time.

Parameters:

`s`

Returns:

True, if the input string represents the same time as this object.

---

equals

```
public boolean equals(java.lang.Object other)
```

This method compares this object with Asn1Time class instance or with Calendar instance. The behavior of this method may be different for different inherited Asn1Time classes. It will return false for all uninitialized time types: so new Asn1UTCTime().equals(new Asn1UTCTime()) will return false.

Parameters:

`other` - Object (instance of Asn1Time or Calendar) to be compared.

Returns:

true, if objects are equal. Uninitialized times are not equal.

---

compareTo

```
public int compareTo(java.lang.Object other)
```

This method compares this object with Asn1Time class instance or with Calendar 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:

(continued from last page)

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

parseString

```
public abstract void parseString(java.lang.String string)
  throws Asn1Exception
```

This method parses passed time string. Note that the action of this method may differentiate for different inherited Asn1Time classes.

Parameters:

`string` - The time string value to be parsed.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

compileString

```
protected abstract boolean compileString()
  throws Asn1Exception
```

Compiles new time string accoring X.680 (clauses 41, 42) and ISO 8601.

Returns:

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

Throws:

`Asn1Exception` - Thrown, if operation is failed.

parseInt

```
protected static int parseInt(java.lang.String str,
         IntHolder off,
         int len)
```

Parses integer value using String.

Parameters:

`str` - string is containing integer to be parsed.
`off` - start offset int the String.
`len` - number of digits to be parsed.

Returns:

Parsed integer value.

putInteger

```
public static void putInteger(java.lang.StringBuffer data,
         int width,
         int value)
```

Puts integer in string buffer

Parameters:

`data` - destination buffer
`width` - number of digits
`value` - value to be put

putInteger

```
protected void putInteger(int width,
         int value)
```

(continued from last page)

Puts integer in string buffer

Parameters:

`width` - number of digits
`value` - value to be put

charAt

```
protected static char charAt(java.lang.String s,
         int index)
```

Returns the character at the specified index in the specified string. If index is out of bounds, then " character will be returned.

Returns:

Character at the specified index. or ".

safeParseString

```
protected void safeParseString()
```

decode

```
protected void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
protected int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done
`tag` - Universal tag to apply

Returns:

Length in octets of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

encode

```
public void encode(Asn1BerOutputStream out,
          boolean explicit,
          Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done
`tag` - Universal tag to apply

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes and writes to stream an ASN.1 time string value using the standard Packed Encoding Rules (PER) encode method.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

decodeXML

```
public void decodeXML(java.lang.String buffer,
          java.lang.String attrs)
  throws Asn1Exception
```

This method decodes ASN.1 GeneralizedTime type, using the XML schema encoding rules.

Parameters:

(continued from last page)

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

parseXmlString

```
public void parseXmlString(java.lang.String string)
  throws Asn1Exception
```

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:

`string` - The time string value to be parsed.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes this ASN.1 time into xsd:dateTime format with element and attribute name tag according to the XML Encoding as specified in the XML schema standard(asn2xsd). This is for use with Obj-Sys XML encoding rules.

Parameters:

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

encodeXER

```
public void encodeXER(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes this ASN.1 time according to XER encoding rules. It is used by generated code when compiling for extended-XER.

Parameters:

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

encodeXMLData

```
public void encodeXMLData(Asn1XmlXerEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes this ASN.1 time string into xsd:dateTime format. XML dateTime format is YYYY-MM-DDTHH:MM:SS[.SSSS][(Z|(+|-) HH:MM)]

Parameters:

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

com.objsys.asn1j.runtime
Class Asn1ArrayType

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1ArrayType
```

public class **Asn1ArrayType**
extends Asn1Type

This class represents the temporaray sequence of XSD type.

Field Summary

java.util.ArrayList	element This member variable is where the list value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1ArrayType() The default constructor initializes the choiceID and value.
public	Asn1ArrayType(Asn1Type[] objects) This constructor creates a an internal array using the given array of objects derived from Asn1Type.

Method Summary

void	add(Asn1Type data)
boolean	equals(java.lang.Object cv_) This method compares this type element with the passed type element.
int	hashCode() This method will return the hashCode for this array type.
void	print(java.io.PrintWriter _out,java.lang.String _varName,int _level) This method will format and output a primitive value to the given print stream, for each array element value.

void	print(java.lang.StringBuilder _sb,java.lang.String _varName,int _level) This method will format and output a primitive value to the given string builder for each array value.
int	size() This method returns the size of array.
Asn1Type[]	toArray() This method returns the array of element object.
void	toArray(Asn1Type[] target) This method copies the elements into the given array.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

element

public transient java.util.ArrayList **element**

This member variable is where the list value is stored.

Constructors

Asn1ArrayType

public **Asn1ArrayType**()

The default constructor initializes the choiceID and value.

Asn1ArrayType

public **Asn1ArrayType**(Asn1Type[] objects)

This constructor creates a an internal array using the given array of objects derived from Asn1Type.

Methods

add

public void **add**(Asn1Type data)

size

`public int size()`

This method returns the size of array.

toArray

`public Asn1Type[] toArray()`

This method returns the array of element object.

toArray

`public void toArray(Asn1Type[] target)`

This method copies the elements into the given array. The given array should be at least sufficiently large to hold all of the elements.

equals

`public boolean equals(java.lang.Object cv_)`

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

Parameters:

`cv_` - Asn1ArrayType value

hashCode

`public int hashCode()`

This method will return the hashCode for this array type.

print

```
public void print(java.io.PrintWriter _out,
          java.lang.String _varName,
          int _level)
```

This method will format and output a primitive value to the given print stream, for each array element value.

Parameters:

`_out` - Print output stream
`_varName` - Name of variable
`_level` - Indentation level

print

```
public void print(java.lang.StringBuilder _sb,
          java.lang.String _varName,
          int _level)
```

This method will format and output a primitive value to the given string builder for each array value. Synchronization is left to the caller.

Parameters:

`_sb` - The output StringBuilder.
`_varName` - The name of the variable.
`_level` - The indentation level.

com.objsys.asn1j.runtime
Class Asn1Base64BinaryString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
         ┗-com.objsys.asn1j.runtime.Asn1Base64BinaryString
```

public class **Asn1Base64BinaryString**
extends Asn1OctetString

This is a container class for holding the components of an XML Base64Binary value.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OctetString`
`TAG`, `value`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`BIT_STRING`, `BMPString`, `BOOLEAN`, `DATE`, `DATE_TIME`, `DURATION`, `ENUMERATED`, `EOC`, `EXTERNAL`, `GeneralString`, `GeneralTime`, `GraphicString`, `IA5String`, `INTEGER`, `mNonParameterizedTypeName`, `NULL`, `NumericString`, `OBJECT_IDENTIFIER`, `ObjectDescriptor`, `OCTET_STRING`, `OID_IRI`, `OpenType`, `PrintableString`, `REAL`, `RELATIVE_OID_IRI`, `RelativeOID`, `SEQUENCE`, `SET`, `T61String`, `TeletexString`, `TIME`, `TIME_OF_DAY`, `UniversalString`, `UTCTime`, `UTF8String`, `VideotexString`, `VisibleString`

Constructor Summary

public	`Asn1Base64BinaryString()` This constructor creates an empty octet string that can be used in a decode method call to receive an octet string value.
public	`Asn1Base64BinaryString(byte[] data)` This constructor initializes an octet string from the given byte array.
public	`Asn1Base64BinaryString(byte[] data,int offset,int nbytes)` This constructor initializes an octet string from a portion of the given byte array.
public	`Asn1Base64BinaryString(java.lang.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.

Method Summary

void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes a base64 encoded character string type into binary octets.

void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix) This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName) This method encodes a base64 string type attribute using the XML encoding as specified in the XML schema standard(asn2xsd).
java.lang.String	toString() This method will return a string representation of the octet string value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1OctetString

compareTo, decode, decode, decode, decode, decode, decode, decode, decode, decodeAsBase64, decodeAsHex, decodeContent, decodeRemainingBits, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsBase64, encodeAsHex, encodeAttribute, encodeBase64Binary, encodeContent, equals, equals, equals, getAsn1TypeName, getLength, getMderLength, hashCode, toInputStream, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1Base64BinaryString

public **Asn1Base64BinaryString**()

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

Asn1Base64BinaryString

public **Asn1Base64BinaryString**(byte[] data)

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

Parameters:

data - Byte array containing an octet string in binary form.

(continued from last page)

Asn1Base64BinaryString

```
public Asn1Base64BinaryString(byte[] data,
                              int offset,
                              int nbytes)
```

This constructor initializes an octet string from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

Parameters:

`data` - Byte array containing an octet string in binary form.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array

Asn1Base64BinaryString

```
public Asn1Base64BinaryString(java.lang.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

Methods

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes a base64 encoded character string type into binary octets.

Parameters:

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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 prefix value

---

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes a base64 string type attribute using the XML encoding as specified in the XML schema standard(asn2xsd).

(continued from last page)

Parameters:

`buffer` - Encode message buffer object
`attrName` - Attribute name

toString

`public java.lang.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

com.objsys.asn1j.runtime
Class Asn1BCDString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
         ┗-com.objsys.asn1j.runtime.Asn1BCDString
```

public class **Asn1BCDString**
extends Asn1OctetString

Asn1BCDString represents an OCTET STRING with an overridden toString method that converts the bytes to a character string of 0-9,A-E characters.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OctetString`
`TAG`, `value`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`BIT_STRING`, `BMPString`, `BOOLEAN`, `DATE`, `DATE_TIME`, `DURATION`, `ENUMERATED`, `EOC`, `EXTERNAL`, `GeneralString`, `GeneralTime`, `GraphicString`, `IA5String`, `INTEGER`, `mNonParameterizedTypeName`, `NULL`, `NumericString`, `OBJECT_IDENTIFIER`, `ObjectDescriptor`, `OCTET_STRING`, `OID_IRI`, `OpenType`, `PrintableString`, `REAL`, `RELATIVE_OID_IRI`, `RelativeOID`, `SEQUENCE`, `SET`, `T61String`, `TeletexString`, `TIME`, `TIME_OF_DAY`, `UniversalString`, `UTCTime`, `UTF8String`, `VideotexString`, `VisibleString`

Constructor Summary

`public`	`Asn1BCDString()` This constructor creates an empty octet string that can be used in a decode method call to receive an octet string value.
`public`	`Asn1BCDString(byte[] data)` This constructor initializes an octet string from the given byte array.
`public`	`Asn1BCDString(byte[] data,int offset,int nbytes)` This constructor initializes an octet string from a portion of the given byte array.
`public`	`Asn1BCDString(java.lang.String value_)` This constructor parses the given BCD character string and assigns the value to the internal octet string.

Method Summary

`java.lang.String`	`toString()` This method will return a string representation of the octet string, using Asn1Util.BCDToString.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1OctetString`

compareTo, decode, decode, decode, decode, decode, decode, decode, decode, decodeAsBase64, decodeAsHex, decodeContent, decodeRemainingBits, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsBase64, encodeAsHex, encodeAttribute, encodeBase64Binary, encodeContent, equals, equals, equals, getAsn1TypeName, getLength, getMderLength, hashCode, toInputStream, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1BCDString

`public` **`Asn1BCDString`**`()`

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

Asn1BCDString

`public` **`Asn1BCDString`**`(byte[] data)`

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

Parameters:

`data` - Byte array containing an octet string in binary form.

Asn1BCDString

```
public Asn1BCDString(byte[] data,
                     int offset,
                     int nbytes)
```

This constructor initializes an octet string from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

Parameters:

`data` - Byte array containing an octet string in binary form.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array

Asn1BCDString

`public` **`Asn1BCDString`**`(java.lang.String value_)`

This constructor parses the given BCD character string and assigns the value to the internal octet string.

Parameters:

(continued from last page)

`value_` - BCD character string. E.g. "5551212E"

Methods

toString

`public java.lang.String` **`toString`**`()`

This method will return a string representation of the octet string, using Asn1Util.BCDToString.

com.objsys.asn1j.runtime
Class Asn1BerDecodeBuffer

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |-com.objsys.asn1j.runtime.Asn1MessageBuffer
         |-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            |-com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
```

Direct Known Subclasses:
Asn1DerDecodeBuffer, Asn1BerInputStream

---

public class **Asn1BerDecodeBuffer**
extends Asn1DecodeBuffer

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1BerDecodeBuffer(byte[] msgdata)` This constructor creates a BER decode buffer object that references an encoded ASN.1 message.
public	`Asn1BerDecodeBuffer(java.io.InputStream istream)` This constructor creates a BER decode buffer object that references an encoded ASN.1 message.

Method Summary

static int	`calcIndefLen(byte[] data,int offset,int len)` This function calculates the actual length of an indefinite length message component.
int	`decodeEnumValue(Asn1Tag tag,boolean explicit,int implicitLength)` This method decodes an enumerated value from the buffer.
int	`decodeEnumValue(boolean explicit,int implicitLength)` This method decodes an enumerated value from the buffer.

int	`decodeLength()` This method decodes a length value.
byte[]	`decodeOpenType()` This method decodes an ASN.1 BER open type value.
byte[]	`decodeOpenType(boolean saveData)` This method decodes an ASN.1 BER open type value.
void	`decodeTag(Asn1Tag tag)` This method decodes a tag value.
int	`decodeTagAndLength(Asn1Tag tag)` This method decodes a tag and length value.
Asn1Tag	`getLastTag()` This method will return the last tag parsed within this decode buffer object.
boolean	`matchTag(Asn1Tag tag)` This overloaded version of matchTag will just test for a match and not return parsed tag and length values
boolean	`matchTag(Asn1Tag tag,Asn1Tag parsedTag,IntHolder parsedLen)` This overloaded version of matchTag allows the tag value to be matched to be passed using an Asn1Tag object.
boolean	`matchTag(short tagClass,short tagForm,int tagIDCode,Asn1Tag parsedTag,IntHolder parsedLen)` This method decodes the next tag value and checks for a match with the given tag value.
void	`movePastEOC(boolean saveData)`
void	`parse(Asn1TaggedEventHandler handler)` This method parses the complete message and invokes the event handler callback methods as various items are encountered.
Asn1Tag	`peekTag()` This overloaded version of the peekTag method will return a reference to a newly created tag object.
void	`peekTag(Asn1Tag parsedTag)` This method will parse and return the next tag in the decode stream without advancing the decode cursor.
int	`readByte()` This method returns the next available 8-bit value from the input stream.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`

`addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBuffer`

addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1BerDecodeBuffer

```
public Asn1BerDecodeBuffer(byte[] msgdata)
```

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

Parameters:

`msgdata` - Byte array containing an encoded ASN.1 message.

Asn1BerDecodeBuffer

```
public Asn1BerDecodeBuffer(java.io.InputStream istream)
```

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

Parameters:

`istream` - Input stream containing an encoded ASN.1 message.

Methods

calcIndefLen

```
public static int calcIndefLen(byte[] data,
         int offset,
         int len)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`data` - Buffer with the indefinite length message component.
`offset` - Offset in the buffer
`len` - Length of the buffer (begining from the offset)

Returns:

calculated length

Throws:

`java.io.IOException` - for I/O exception

decodeLength

```
public final int decodeLength()
  throws Asn1Exception,
         java.io.IOException
```

This method decodes a length value.

Returns:

Decoded length value

Throws:

`java.io.IOException` - for I/O exception

decodeOpenType

```
public byte[] decodeOpenType()
  throws Asn1Exception,
         java.io.IOException
```

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

Returns:

Reference to byte array containing component.

Throws:

`java.io.IOException` - for I/O exception

decodeOpenType

```
public byte[] decodeOpenType(boolean saveData)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`saveData` - True if data should be captured and returned

Returns:

Reference to byte array containing component.

Throws:

`java.io.IOException` - for I/O exception

movePastEOC

```
protected final void movePastEOC(boolean saveData)
  throws Asn1Exception,
         java.io.IOException
```

decodeTag

```
public final void decodeTag(Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes a tag value.

(continued from last page)

Parameters:
`tag` - Tag object to receive decoded tag fields.

Throws:
`java.io.IOException` - for I/O exception

---

decodeTagAndLength

```
public final int decodeTagAndLength(Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes a tag and length value.

Parameters:
`tag` - Tag object to receive decoded tag fields.

Returns:
Decoded length value.

---

getLastTag

```
public Asn1Tag getLastTag()
```

This method will return the last tag parsed within this decode buffer object.

Returns:
Last parsed tag object reference

---

matchTag

```
public final boolean matchTag(short tagClass,
          short tagForm,
          int tagIDCode,
          Asn1Tag parsedTag,
          IntHolder parsedLen)
  throws Asn1Exception,
         java.io.IOException
```

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

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

Returns:
True if given tag matches tag at decode cursor

---

matchTag

```
public final boolean matchTag(Asn1Tag tag,
          Asn1Tag parsedTag,
          IntHolder parsedLen)
  throws Asn1Exception,
         java.io.IOException
```

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

(continued from last page)

Parameters:

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

Returns:

True if given tag matches tag at decode cursor

---

matchTag

```
public final boolean matchTag(Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`tag` - Tag value to be matched.

Returns:

True if given tag matches tag at decode cursor

---

peekTag

```
public final void peekTag(Asn1Tag parsedTag)
  throws Asn1Exception,
         java.io.IOException
```

This method will parse and return the next tag in the decode stream without advancing the decode cursor.

Parameters:

`parsedTag` - Holder object to receive parsed tag value

---

peekTag

```
public final Asn1Tag peekTag()
  throws Asn1Exception,
         java.io.IOException
```

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

Returns:

Parsed tag object value reference

---

parse

```
public void parse(Asn1TaggedEventHandler handler)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`handler` - Object implementing the Asn1EventHandler interface.

---

readByte

```
public int readByte()
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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

Returns:

Next 8-bit byte value from input stream

---

decodeEnumValue

```
public int decodeEnumValue(boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an enumerated value from the buffer. This assumes that the enumerated value is tagged [UNIVERSAL 10].

Parameters:

`explicit -` Flag that indicates element is explicitly tagged.

`implicitLength -` Length of contents if implicit.

Returns:

The decoded integer value.

---

decodeEnumValue

```
public int decodeEnumValue(Asn1Tag tag,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an enumerated value from the buffer. This method requires an explicit tag to be passed for decoding.

Parameters:

`tag -` The tag to match for decoding.

`explicit -` Flag that indicates element is explicitly tagged.

`implicitLength -` Length of contents if implicit.

Returns:

The decoded integer value.

com.objsys.asn1j.runtime
Class Asn1BerDecodeContext

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1BerDecodeContext
```

public class **Asn1BerDecodeContext**
extends java.lang.Object

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

Field Summary

int	`mDecBufByteCount` This variable is used to keep track of the current byte count in the decode buffer.
Asn1BerDecodeBuffer	`mDecodeBuffer` This variable holds a reference to the BER decode buffer object that is being used to decode the entire message component.
int	`mElemLength` This variable holds the contructed element length for the context component.
boolean	`mExplicitTagging` This boolean flag variable indicates if explicit tagging is in effect for this element.
java.util.Vector	`mTagHistory` The tag history is a list of decoded tags.
Asn1Tag	`mTagHolder` This variable holds the current parsed tag for matching operations.

Constructor Summary

public	`Asn1BerDecodeContext(Asn1BerDecodeBuffer decodeBuffer,int elemLength)` The constructor initializes all internal working varaibles.

Method Summary

void	`addTag(Asn1Tag tag)` This method adds a tag to the history.
boolean	`expired()` This method will determine if a decoding context is expired.
boolean	`hasDecoded(Asn1Tag tag)` This method checks to see whether a tag has been already decoded.

boolean	`matchElemTag(Asn1Tag tag,IntHolder parsedLen,boolean advance)` This method will attempt to match the next element tag in a constructed type with the expected value.
boolean	`matchElemTag(short tagClass,short tagForm,int tagIDCode,IntHolder parsedLen,boolean advance)` This method will attempt to match the next element tag in a constructed type with the expected value.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

mDecodeBuffer

`protected com.objsys.asn1j.runtime.Asn1BerDecodeBuffer` **`mDecodeBuffer`**

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

mDecBufByteCount

`protected int` **`mDecBufByteCount`**

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

mElemLength

`protected int` **`mElemLength`**

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

mExplicitTagging

`protected boolean` **`mExplicitTagging`**

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

mTagHolder

`protected com.objsys.asn1j.runtime.Asn1Tag` **`mTagHolder`**

This variable holds the current parsed tag for matching operations.

mTagHistory

`protected java.util.Vector` **`mTagHistory`**

The tag history is a list of decoded tags. This is used when decoding SETs to ensure that duplicate elements are not encoded.

Constructors

(continued from last page)

Asn1BerDecodeContext

```
public Asn1BerDecodeContext(Asn1BerDecodeBuffer decodeBuffer,
                            int elemLength)
```

The constructor initializes all internal working varaibles.

Parameters:

`decodeBuffer` - Reference to current decode buffer method.
`elemLength` - Length of the element being tracked.

Methods

expired

```
public final boolean expired()
  throws Asn1Exception,
         java.io.IOException
```

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

Returns:

True if at the end of the context block

Throws:

`java.io.IOException` - for I/O exception

---

matchElemTag

```
public final boolean matchElemTag(short tagClass,
          short tagForm,
          int tagIDCode,
          IntHolder parsedLen,
          boolean advance)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

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

Returns:

True if at the end of the context block

Throws:

`java.io.IOException` - for I/O exception

(continued from last page)

matchElemTag

```
public final boolean matchElemTag(Asn1Tag tag,
          IntHolder parsedLen,
          boolean advance)
  throws Asn1Exception,
          java.io.IOException
```

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

Parameters:

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

Returns:

True if at the end of the context block

Throws:

`java.io.IOException` - for I/O exception

---

addTag

```
public final void addTag(Asn1Tag tag)
```

This method adds a tag to the history.

Parameters:

`tag` - The tag to be added to the history.

---

hasDecoded

```
public final boolean hasDecoded(Asn1Tag tag)
```

This method checks to see whether a tag has been already decoded.

Parameters:

`tag` - The tag to be searched.

Returns:

True if the tag is in the history.

com.objsys.asn1j.runtime
Interface Asn1BerDecoder

public interface **Asn1BerDecoder**

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

Method Summary

`abstract Asn1Type`	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` Decode value from given buffer.

Methods

decode

```
public abstract Asn1Type decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

Decode value from given buffer.

Parameters:

`buffer` - The buffer to decode from.
`explicit` - if `true`, the value to be decoded is preceded by a tag and length, which must first be decoded. Otherwise, implicitLength provides the length of the value to be decoded.
`implicitLength` - The length of the value to be decoded if explicit was given as `false`.

Returns:

The decoded type.

Throws:

`java.io.IOException` - for I/O exception

com.objsys.asn1j.runtime
Class Asn1BerEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev
               ┗-com.objsys.asn1j.runtime.Asn1BerEncodeBuffer
```

Direct Known Subclasses:
Asn1DerEncodeBuffer

---

public class **Asn1BerEncodeBuffer**
extends Asn1EncodeByteBufferRev

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev`
`mByteIndex`, `mData`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`
`INITIAL_SIZE`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1BerEncodeBuffer()` This constructor creates a BER encode buffer object with the default initial size.
public	`Asn1BerEncodeBuffer(int size)` This constructor creates a BER encode buffer object with the given initial size.

Method Summary

void	`binDump()`

void	binDump(java.io.PrintStream out,java.lang.String varName) This method dumps the encoded message in a human-readable format showing tags and contents to the given print output stream.
void	checkSize(int bytesRequired)
int	encodeIdentifier(int ident) This method encodes an ASN.1 identifier value such as the ones used in a tags or object identifiers.
int	encodeIntValue(long ivalue) This method encodes an ASN.1 integer value's contents according to the ASN.1 Basic Encoding Rules (BER)..
int	encodeLength(int len) This method encodes a length value.
int	encodeTag(Asn1Tag tag) This method encodes a tag value.
int	encodeTag(short tagClass,short tagForm,int tagIdent) This method encodes a tag value.
int	encodeTagAndLength(Asn1Tag tag,int len) This method encodes both a tag and length value.
int	encodeTagAndLength(short tagClass,short tagForm,int tagIDCode,int len) This overloaded version of encodeTagAndLength allows tag value components to be specified instead of an Asn1Tag object
boolean	isBER() This method is used for testing encode rules type.

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev
checkSize, copy, copy, copy, copy, getByteArrayInputStream, getMsgCopy, getMsgLength, initBuffer, reset, toString, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1BerEncodeBuffer

public **Asn1BerEncodeBuffer**()

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

Asn1BerEncodeBuffer

public **Asn1BerEncodeBuffer**(int size)

This constructor creates a BER encode buffer object with the given initial size. Whenever the buffer becomes full, it will be expanded. For best performance, this size should be large enough to prevent resizing in normal operation.

Parameters:

size - The initial size in bytes of an encode buffer.

Methods

checkSize

protected void **checkSize**(int bytesRequired)

encodeIdentifier

public int **encodeIdentifier**(int ident)

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

Parameters:

ident - The identifier to be encoded.

Returns:

Length of the encoded component in octets.

encodeIntValue

public int **encodeIntValue**(long ivalue)

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

Parameters:

ivalue - Integer value to encode

Returns:

Length of encoded component

(continued from last page)

encodeLength

```
public int encodeLength(int len)
```

This method encodes a length value.

Parameters:

`len` - The length to be encoded.

Returns:

Length of encoded component

---

encodeTag

```
public int encodeTag(Asn1Tag tag)
```

This method encodes a tag value.

Parameters:

`tag` - The tag to be encoded.

Returns:

Length of component or negative status value

---

encodeTag

```
public int encodeTag(short tagClass,
                     short tagForm,
                     int tagIdent)
```

This method encodes a tag value.

Parameters:

`tagClass` - Tag class value (UNIV, APPL, CTXT, or PRIV)
`tagForm` - Tag form value (PRIM or CONS)
`tagIdent` - Tag identifer code

Returns:

Length of component or negative status value

---

encodeTagAndLength

```
public int encodeTagAndLength(Asn1Tag tag,
                              int len)
```

This method encodes both a tag and length value.

Parameters:

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

Returns:

Length of encoded component

---

encodeTagAndLength

```
public int encodeTagAndLength(short tagClass,
                              short tagForm,
                              int tagIDCode,
                              int len)
```

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

(continued from last page)

Parameters:

`tagClass` - The class of the tag to be encoded.
`tagForm` - The form of the tag to be encoded.
`tagIDCode` - The ID code of the tag to be encoded.
`len` - The length to be encoded.

Returns:

status value (see Asn1Status.java)

---

binDump

```
public void binDump(java.io.PrintStream out,
         java.lang.String varName)
```

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

---

binDump

```
public void binDump()
```

---

isBER

```
public boolean isBER()
```

This method is used for testing encode rules type.

Returns:

true

com.objsys.asn1j.runtime
Class Asn1BerInputStream

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
               ┗-com.objsys.asn1j.runtime.Asn1BerInputStream
```

All Implemented Interfaces:
Asn1InputStream

Direct Known Subclasses:
Asn1CerInputStream

---

public class **Asn1BerInputStream**
extends Asn1BerDecodeBuffer
implements Asn1InputStream

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1BerInputStream(java.io.InputStream istream)` This constructor creates a BER input stream object that references an encoded ASN.1 message.

Method Summary

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.
void	`close()` Closes this input stream and releases any system resources associated with the stream.

void	`mark(int readLimit)` This method is used to mark the current position in the input stream for retry processing.
boolean	`markSupported()` Tests if this input stream supports the `mark`and `reset`methods.
void	`reset()` This method is used to reset the current position in the input stream back to the location of the last 'mark' call.
long	`skip(long nbytes)` This method will skip over the requested number of bytes in the input stream.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1BerDecodeBuffer`

`calcIndefLen, decodeEnumValue, decodeEnumValue, decodeLength, decodeOpenType, decodeOpenType, decodeTag, decodeTagAndLength, getLastTag, matchTag, matchTag, matchTag, movePastEOC, parse, peekTag, peekTag, readByte`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`

`addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBuffer`

`addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`

`getContext, hexDump, hexDump, setKey, setTypeCode`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1BerInputStream

`public` **`Asn1BerInputStream`**`(java.io.InputStream istream)`

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

Parameters:

`istream` - Input stream containing an encoded ASN.1 message.

Methods

(continued from last page)

available

```
public int available()
  throws java.io.IOException
```

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.

Throws:

`IOException` - if an I/O error occurs.

---

close

```
public void close()
  throws java.io.IOException
```

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

Throws:

`IOException` - if an I/O error occurs.

---

mark

```
public void mark(int readLimit)
```

This method is used to mark the current position in the input stream for retry processing. It is equivalent to the Java InputStream 'mark' method.

Parameters:

`readLimit` - Max number of bytes that can be read before mark becomes invalid.

---

markSupported

```
public boolean markSupported()
```

Tests if this input stream supports the `mark`and `reset`methods. The `markSupported`method of `InputStream`returns `false`.

Returns:

`true`if this true type supports the mark and reset method; `false`otherwise.

---

reset

```
public 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 the Java InputStream 'reset' method.

---

skip

```
public long skip(long nbytes)
  throws java.io.IOException
```

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

Parameters:

`nbytes` - Number of bytes to skip

Throws:

(continued from last page)

`IOException` - if an I/O error occurs.

com.objsys.asn1j.runtime
Class Asn1BerMessageDumpHandler

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1BerMessageDumpHandler
```

All Implemented Interfaces:
Asn1TaggedEventHandler

---

public class **Asn1BerMessageDumpHandler**
extends java.lang.Object
implements Asn1TaggedEventHandler

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

Constructor Summary

public	`Asn1BerMessageDumpHandler(java.io.PrintStream out)` The constructor sets the PrintStream object to which the formatted output should be written.

Method Summary

void	`contents(byte[] data)` This method is invoked after each contents field is parsed.
void	`endElement(Asn1Tag tag)` This method is invoked after parsing is complete on each tag/length/value (TLV) in the message.
void	`startElement(Asn1Tag tag,int len,byte[] tagLenBytes)` This method is invoked after each tag/length value is parsed in the message being dumped.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1BerMessageDumpHandler

`public` **`Asn1BerMessageDumpHandler`**`(java.io.PrintStream out)`

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

Parameters:

(continued from last page)

`out` - Output stream for formatted data

Methods

startElement

```
public void startElement(Asn1Tag tag,
         int len,
         byte[] tagLenBytes)
```

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

Parameters:

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

---

contents

```
public void contents(byte[] data)
```

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

Parameters:

`data` - Array containing the encoded contents bytes

---

endElement

```
public void endElement(Asn1Tag tag)
```

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

Parameters:

`tag` - Array containing the encoded contents bytes

com.objsys.asn1j.runtime
Class Asn1BerOutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-com.objsys.asn1j.runtime.Asn1OutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1BerOutputStream
```

Direct Known Subclasses:
Asn1CerOutputStream

---

public class **Asn1BerOutputStream**
extends Asn1OutputStream

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OutputStream`
`os`

Constructor Summary

public	`Asn1BerOutputStream(java.io.OutputStream os)` This constructor creates a buffered BER output stream object with default size of buffer.
public	`Asn1BerOutputStream(java.io.OutputStream os,int bufSize)` This constructor creates a buffered BER output stream object.

Method Summary

void	`encode(Asn1Type object,boolean explicit)` This method encodes and writes to the stream ASN.1 types.
void	`encodeBitString(byte[] value,int numbits,boolean explicit,Asn1Tag tag)` This method writes the given array of bytes as bit string value.
void	`encodeBMPString(java.lang.String value,boolean explicit,Asn1Tag tag)` This method writes the given string as BMP string value.
void	`encodeCharString(java.lang.String value,boolean explicit,Asn1Tag tag)` This method encodes and writes to the stream an ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
void	`encodeEOC()` This method encodes and writes an End-Of-Contents marker to the stream.

void	`encodeIdentifier(long ident)` This method encodes and writes to the stream an ASN.1 identifier value such as those used in tags or object identifiers.
void	`encodeIntValue(long value,boolean encodeLen)` This method encodes and writes to the stream an ASN.1 integer value's contents according to the ASN.1 Basic Encoding Rules (BER).
void	`encodeLength(int len)` This method encodes and writes a length value to the stream.
void	`encodeOctetString(byte[] value,boolean explicit,Asn1Tag tag)` This method writes the given array of bytes as octet string value.
void	`encodeTag(Asn1Tag tag)` This method encodes and writes a tag value to the stream.
void	`encodeTag(short tagClass,short tagForm,int tagIDCode)` This method encodes and writes a tag value to the stream.
void	`encodeTagAndIndefLen(Asn1Tag tag)` This method encodes and writes both a tag and an indefinite length indicator to the stream.
void	`encodeTagAndIndefLen(short tagClass,short tagForm,int tagIDCode)` This overloaded version of encodeTagAndIndefLen allows tag value components to be specified instead of an Asn1Tag object.
void	`encodeTagAndLength(Asn1Tag tag,int len)` This method encodes and writes both a tag and length value to the stream.
void	`encodeUnivString(int[] value,boolean explicit,Asn1Tag tag)` This method writes the given array of integers as UniversalString value.
boolean	`isBER()` This method is used for testing encode rules type.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1OutputStream`
`close, flush, getContext, write, write, write, write2Bytes, write4Bytes`

Methods inherited from class `java.io.OutputStream`
`close, flush, write, write, write`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1BerOutputStream

```
public Asn1BerOutputStream(java.io.OutputStream os)
```

This constructor creates a buffered BER output stream object with default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream. Note: in a Java ME environment, the stream will not be buffered.

Parameters:

`os` - The underlying OutputStream object.

Asn1BerOutputStream

```
public Asn1BerOutputStream(java.io.OutputStream os,
                           int bufSize)
```

This constructor creates a buffered BER output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream. Note: in Java ME environment, the stream will not be buffered.

Parameters:

`os` - The underlying OutputStream object.

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

Methods

encodeIdentifier

```
public void encodeIdentifier(long ident)
  throws java.io.IOException
```

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

Parameters:

`ident` - The identifier to be encoded.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

---

encodeTag

```
public void encodeTag(Asn1Tag tag)
  throws java.io.IOException
```

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

Parameters:

`tag` - The tag to be encoded.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

---

encodeTag

```
public void encodeTag(short tagClass,
         short tagForm,
         int tagIDCode)
  throws java.io.IOException
```

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

Parameters:

`tagClass` - The class of the tag to be encoded.

(continued from last page)

`tagForm` - The form of the tag to be encoded.
`tagIDCode` - The ID code of the tag to be encoded.

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.

---

encodeLength

```
public void encodeLength(int len)
  throws java.io.IOException
```

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

Parameters:
`len` - The length to be encoded.

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.

---

encodeTagAndLength

```
public void encodeTagAndLength(Asn1Tag tag,
         int len)
  throws java.io.IOException
```

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

Parameters:
`tag` - The tag to be encoded.
`len` - The length to be encoded.

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.

---

encodeTagAndIndefLen

```
public void encodeTagAndIndefLen(Asn1Tag tag)
  throws java.io.IOException
```

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

Parameters:
`tag` - The tag to be encoded.

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.

---

encodeTagAndIndefLen

```
public void encodeTagAndIndefLen(short tagClass,
         short tagForm,
         int tagIDCode)
  throws java.io.IOException
```

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

Parameters:
`tagClass` - The class of the tag to be encoded.
`tagForm` - The form of the tag to be encoded.
`tagIDCode` - The ID code of the tag to be encoded.

(continued from last page)

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeEOC

```
public void encodeEOC()
  throws java.io.IOException
```

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeIntValue

```
public void encodeIntValue(long value,
         boolean encodeLen)
  throws java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeBMPString

```
public void encodeBMPString(java.lang.String value,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

This method writes the given string as BMP string value.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encodeBitString

```
public void encodeBitString(byte[] value,
         int numbits,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`value` - Byte array containing data to encode.
`numbits` - Number of bits to encode
`explicit` - Flag indicating explicit tagging should be done

(continued from last page)

`tag` - Universal tag to apply

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeCharString

```
public void encodeCharString(java.lang.String value,
          boolean explicit,
          Asn1Tag tag)
  throws Asn1Exception,
          java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeOctetString

```
public void encodeOctetString(byte[] value,
          boolean explicit,
          Asn1Tag tag)
  throws Asn1Exception,
          java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeUnivString

```
public void encodeUnivString(int[] value,
          boolean explicit,
          Asn1Tag tag)
  throws Asn1Exception,
          java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1Type object,
                   boolean explicit)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`object` - The object to be written

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

isBER

```
public boolean isBER()
```

This method is used for testing encode rules type.

Returns:

true if this is BER, false if CER or DER.

com.objsys.asn1j.runtime
Class Asn1BigDecimal

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Real
         ┗-com.objsys.asn1j.runtime.Asn1BigDecimal
```

public class **Asn1BigDecimal**
extends Asn1Real

This class represents the Xml Schema BigDecimal built-in type.

Field Summary

java.math.BigDecimal	value This public member variable is where the double value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Real
ANY_BASE, ANY_BINARY, BASE_2_ONLY, TAG, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1BigDecimal() The default constructor sets the double value to zero.
public	Asn1BigDecimal(double value_) This constructor creates an REAL object from a double value.
public	Asn1BigDecimal(java.math.BigDecimal value_) This constructor creates an REAL object from a double value.

Method Summary

void	decodeXML(java.lang.String buffer,java.lang.String attrs) This method decodes an ASN.1 real value using the XML schema encoding rules.

void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an xsd:BigDecimal value using the XML Encoding as specified in the W3C XML schema standard.
void	encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName) This method encodes an ASN.1 real value using the XML Encoding as specified in the W3C XML schema standard(asn2xsd).
java.lang.String	encodeBigDecimal()
boolean	equals(double dvalue) This method compares this REAL value to the given value for equality.
boolean	equals(java.lang.Object o)
java.lang.String	toString() This method will return a string representation of the REAL value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Real

decode, decode, decode, decode, decode, decode, decodeDouble, decodeSingle, decodeV72, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAttribute, encodeDouble, encodeSingle, encodeV72, encodeValue, equals, equals, getAsn1TypeName, hashCode, normalizedRealValueToString, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

value

```
public transient java.math.BigDecimal value
```

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.

Constructors

Asn1BigDecimal

```
public Asn1BigDecimal()
```

(continued from last page)

The default constructor sets the double value to zero.

Asn1BigDecimal

```
public Asn1BigDecimal(double value_)
```

This constructor creates an REAL object from a double value.

Parameters:

`value_` - double value

Asn1BigDecimal

```
public Asn1BigDecimal(java.math.BigDecimal value_)
```

This constructor creates an REAL object from a double value.

Parameters:

`value_` - double value

Methods

encode

```
public void encode(Asn1XmlEncoder buffer,
                   java.lang.String elemName,
                   java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an xsd:BigDecimal value using the XML Encoding as specified in the W3C XML schema standard.

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
                            java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

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

encodeBigDecimal

```
public java.lang.String encodeBigDecimal()
```

decodeXML

```
public void decodeXML(java.lang.String buffer,
                      java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 real value using the XML schema encoding rules.

(continued from last page)

Parameters:

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

equals

`public boolean **equals**(double dvalue)`

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

Parameters:

`dvalue` - REAL test value

equals

`public boolean **equals**(java.lang.Object o)`

toString

`public java.lang.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

com.objsys.asn1j.runtime
Class Asn1BigInteger

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1BigInteger
```

public class **Asn1BigInteger**
extends Asn1Type

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 <isBigInteger/> qualifier is specified in a compiler configuration file.

Field Summary

`static final Asn1Tag`	`TAG` The **TAG**constant describes the universal tag for this data type (UNIVERSAL 2).
`java.math.BigInteger`	`value` The `value`member is public and is an instance of a Java BigInteger object.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`BIT_STRING`, `BMPString`, `BOOLEAN`, `DATE`, `DATE_TIME`, `DURATION`, `ENUMERATED`, `EOC`, `EXTERNAL`, `GeneralString`, `GeneralTime`, `GraphicString`, `IA5String`, `INTEGER`, `mNonParameterizedTypeName`, `NULL`, `NumericString`, `OBJECT_IDENTIFIER`, `ObjectDescriptor`, `OCTET_STRING`, `OID_IRI`, `OpenType`, `PrintableString`, `REAL`, `RELATIVE_OID_IRI`, `RelativeOID`, `SEQUENCE`, `SET`, `T61String`, `TeletexString`, `TIME`, `TIME_OF_DAY`, `UniversalString`, `UTCTime`, `UTF8String`, `VideotexString`, `VisibleString`

Constructor Summary

public	`Asn1BigInteger()` The default constructor sets the big integer value object reference to null.
public	`Asn1BigInteger(java.math.BigInteger value)` This constructor sets the big integer object reference to that of the object passed in.
public	`Asn1BigInteger(java.lang.String value)` This constructor creates a new big integer object and sets it to the string value passed in.
public	`Asn1BigInteger(java.lang.String value,int radix)` This constructor creates a new big integer object and sets it to the number string value passed in using the given radix.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 INTEGER from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode this value as if unconstrained.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,java.math.BigInteger lower,java.math.BigInteger upper)` This method decodes an ASN.1 integer value using Packed Encoding Rules (PER).
void	`decodeSigned(Asn1OerDecodeBuffer buffer)` Decode a variable-size signed integer, encoded with a length, into this object, following OER.
void	`decodeSigned(Asn1OerDecodeBuffer buffer,int octets)` Decode a signed integer (2's complement form), of the given length, into this object.
void	`decodeUnsigned(Asn1OerDecodeBuffer buffer)` Decode a variable-size unsigned integer, encoded with a length, into this object, following OER.
void	`decodeUnsigned(Asn1OerDecodeBuffer buffer,int octets)` Decode an unsigned integer (a binary integer, not 2's complement form), of the given length, into this object.
static java.math.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).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 integer value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 integer value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this ASN.1 INTEGER value to JSON

void	`encode(Asn1OerEncodeBuffer buffer)` Encode this value as if unconstrained.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,java.math.BigInteger lower,java.math.BigInteger upper)` This method encodes an ASN.1 integer value using Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out,java.math.BigInteger lower,java.math.BigInteger upper)` This method encodes an ASN.1 integer value using Packed Encoding Rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 integer value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard.
void	`encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName)` This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard.
void	`encodeSigned(Asn1OerEncodeBuffer buffer)` Encode this value as a variable-size signed integer, with length, according to OER.
void	`encodeSigned(Asn1OerEncodeBuffer buffer,int octets)` Encode integer value as a signed value (2's complement) in the given number of octets.
void	`encodeUnsigned(Asn1OerEncodeBuffer buffer)` Encode this value as a variable-size unsigned integer, with length, according to OER.
void	`encodeUnsigned(Asn1OerEncodeBuffer buffer,int octets)` Encode integer value as an unsigned value (binary integer) in the given number of octets.
static int	`encodeValue(Asn1EncodeBuffer buffer,java.math.BigInteger ivalue,boolean doCopy)`
boolean	`equals(java.lang.Object x)` Compares this Asn1BigInteger with the specified Object for equality.
java.lang.String	`toString()` This method will return a string representation of the integer value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 2).

value

public transient java.math.BigInteger **value**

The valuemember is public and is an instance of a Java BigInteger object.

Constructors

Asn1BigInteger

public **Asn1BigInteger**()

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

Asn1BigInteger

public **Asn1BigInteger**(java.math.BigInteger value)

This constructor sets the big integer object reference to that of the object passed in.

Parameters:

value - BigInteger object reference

Asn1BigInteger

public **Asn1BigInteger**(java.lang.String value)

This constructor creates a new big integer object and sets it to the string value passed in. The string value may contain a prefix for the radix: 0x - hexadecimal, 0o - octal, 0b - binary. none - decimal All values are understood as positive values, unless there is a leading minus sign.

Parameters:

value - String value

Asn1BigInteger

public **Asn1BigInteger**(java.lang.String value,
int radix)

(continued from last page)

This constructor creates a new big integer object and sets it to the number string value passed in using the given radix. The string value should be as for `int)`

Parameters:

`value` - String value

Methods

decodeValue

```
public static java.math.BigInteger decodeValue(Asn1DecodeBuffer buffer,
         int length)
  throws Asn1Exception,
         java.io.IOException
```

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

---

encodeValue

```
protected static int encodeValue(Asn1EncodeBuffer buffer,
         java.math.BigInteger ivalue,
         boolean doCopy)
```

---

equals

```
public boolean equals(java.lang.Object x)
```

Compares this Asn1BigInteger with the specified Object for equality.

Parameters:

`x` - Object to which this Asn1BigInteger is to be compared.

Returns:

trueif and only if the specified Object is a Asn1BigInteger or BigInteger whose value is numerically equal to this Asn1BigInteger.

---

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

(continued from last page)

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of component or negative status value

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable.

Parameters:

`buffer` - PER Decode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         java.math.BigInteger lower,
         java.math.BigInteger upper)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - PER Decode message buffer object

`lower` - The PER-visible lower bound; null if there is not a lower bound.

`upper` - The PER-visible upper bound; null if there is not an upper bound.

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         java.math.BigInteger lower,
         java.math.BigInteger upper)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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

Parameters:

`buffer` - PER Encode message buffer object
`lower` - The PER-visible lower bound; null if there is not a lower bound.
`upper` - The PER-visible upper bound; null if there is not an upper bound.

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode this value as if unconstrained. Subclasses may override this method to decode according to the constraints on the respective ASN.1 type.

Parameters:

`buffer`

---

decodeSigned

```
public final void decodeSigned(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a variable-size signed integer, encoded with a length, into this object, following OER.

---

decodeUnsigned

```
public final void decodeUnsigned(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a variable-size unsigned integer, encoded with a length, into this object, following OER.

---

decodeSigned

```
public final void decodeSigned(Asn1OerDecodeBuffer buffer,
         int octets)
  throws java.io.IOException
```

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

Parameters:

`octets` - The number of octets in which the value was encoded.

---

decodeUnsigned

```
public final void decodeUnsigned(Asn1OerDecodeBuffer buffer,
         int octets)
  throws java.io.IOException
```

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

Parameters:

`octets` - The number of octets in which the value was encoded.

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode this value as if unconstrained. Subclasses may override this method to encode it according to the constraints on the respective ASN.1 type.

Parameters:

(continued from last page)

buffer

---

encodeSigned

```
public final void encodeSigned(Asn1OerEncodeBuffer buffer,
          int octets)
```

Encode integer value as a signed value (2's complement) in the given number of octets. The value must fit in the given number of octets.

Parameters:

`buffer` - The buffer.
`octets` - The number of octets to encode in; 0 < octets <= 8

---

encodeSigned

```
public final void encodeSigned(Asn1OerEncodeBuffer buffer)
```

Encode this value as a variable-size signed integer, with length, according to OER.

---

encodeUnsigned

```
public final void encodeUnsigned(Asn1OerEncodeBuffer buffer)
```

Encode this value as a variable-size unsigned integer, with length, according to OER.

---

encodeUnsigned

```
public final void encodeUnsigned(Asn1OerEncodeBuffer buffer,
          int octets)
```

Encode integer value as an unsigned value (binary integer) in the given number of octets. The value must be non-negative and fit in the given number of octets.

Parameters:

`buffer` - The buffer.
`octets` - The number of octets to encode in; 0 < octets <= 8.

---

encode

```
public void encode(Asn1XerEncoder buffer,
          java.lang.String elemName)
  throws java.io.IOException,
          Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object
`elemName` - Element name

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
          java.lang.String attrs)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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` - The namespace prefix

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

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
`attrName` - Attribute name

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

Parameters:

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

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 INTEGER from JSON.

Parameters:

`buffer`

Throws:

`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this ASN.1 INTEGER value to JSON

(continued from last page)

toString

```
public java.lang.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

---

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
         java.math.BigInteger lower,
         java.math.BigInteger upper)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - PER Encode message output stream

`lower` - The PER-visible lower bound; null if there is not a lower bound.

`upper` - The PER-visible upper bound; null if there is not an upper bound.

com.objsys.asn1j.runtime
Interface Asn1BitMessageBuffer

All Subinterfaces:
Asn1PerMessageBuffer

All Known Implementing Classes:
Asn1EncodeBitBuffer, Asn1DecodeBitBuffer

---

public interface **Asn1BitMessageBuffer**

This interface defines constants and methods common to bit-oriented encoders and decoders.

Method Summary

abstract void	byteAlign() This method handles byte-alignment for aligned PER encoding or decoding.
abstract java.io.InputStream	getInputStream() This method returns an input stream object that represents the message being encoded or decoded.
abstract int	getMsgBitCnt() This method returns the count of bits in the encoded message.
abstract Asn1PerTraceHandler	getTraceHandler() This method returns the internal trace handler object used to trace all of the bit fields with a PER message.

Methods

byteAlign

`public abstract void **byteAlign**()`

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

---

getInputStream

`public abstract java.io.InputStream **getInputStream**()`

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

---

getMsgBitCnt

`public abstract int **getMsgBitCnt**()`

This method returns the count of bits in the encoded message.

---

getTraceHandler

`public abstract Asn1PerTraceHandler **getTraceHandler**()`

(continued from last page)

This method returns the internal trace handler object used to trace all of the bit fields with a PER message.

com.objsys.asn1j.runtime
Class Asn1BitString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1BitString
```

public class **Asn1BitString**
extends Asn1Type

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

Field Summary

static final short	ASN1VALUE The **ASN1VALUE**constant describes the string format for hex or binary digit value. Value: **2**
static final short	BITS The **BITS**constant describes the string format for binary digit value (e.g. Value: **1**
static final short	HEX The **HEX**constant describes the string format for hex digit value (e.g. Value: **0**
static final short	HEXBIN The **HEXBIN**constant describes the string format for hex or binary digit value (e.g. Value: **3**
static short	mStringFormat The **mStringFormat**variable can be used to set the string format for print or event handler or toString() functions.
int	numbits This variable contains the number of bits in the bit string value.
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 3).
boolean	trimZeroBits This variable describes whether trailing zero bits should be truncated.
byte[]	value This variable holds the bit string value.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	`Asn1BitString()` This constructor creates an empty bit string that can be used in a decode method call to receive a bit string value.
public	`Asn1BitString(int numbits_,byte[] data)` This constructor initializes a bit string with the given number of bits and data.
public	`Asn1BitString(byte[] data)` This constructor initializes a bit string with the given bytes, using all 8 bits of every byte.
public	`Asn1BitString(boolean[] bitValues)` This constructor initializes a bit string from the given boolean array.
public	`Asn1BitString(java.lang.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.
public	`Asn1BitString(java.util.BitSet bitSet)` This constructor initializes a bit string from the given BitSet object.

Method Summary

void	`clear(int bitno)` This method clears the given bit in the bit string.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 bit string value using the BER or DER encoding rules.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 BIT STRING from JSON.
void	`decode(Asn1MderDecodeBuffer buffer,int length)` Decode a BIT STRING from the MDER encoding into this object.
void	`decode(Asn1NasDecodeBuffer buffer,int numbits)` Decode the given number of bits from the buffer.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 BIT STRING that was encoded according to OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 bit string value using the packed encoding rules (PER).

void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,java.lang.Object upper)` This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,java.lang.Object lower,long upper)` This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,java.lang.Object lower,java.lang.Object upper)` This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER).
void	`decodeContent(Asn1OerDecodeBuffer buffer,int numOctets,int unusedBits)` This method decodes an ASN.1 BIT STRING's content that was encoded according to OER, given the total number of octets and the number of unused bits in the last octet.
void	`decodeRemainingBits(Asn1NasDecodeBuffer buffer)` Decode all remaining bits in the current container from the buffer.
void	`decodeString(Asn1JsonDecodeBuffer buffer,int numbits)` Decode ASN.1 BIT STRING from JSON.
void	`decodeV72(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 bit string from JSON.
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 BIT STRING using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 bit string value using the BER or DER encoding rules.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode ASN.1 BIT STRING to JSON.
void	`encode(Asn1MderOutputStream out,int length)` Encode this BIT STRING into the MDER encoding.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes this ASN.1 BIT STRING value, according to OER.

void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an unconstrained ASN.1 bit string value using the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,java.lang.Object upper)` This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,java.lang.Object lower,long upper)` This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,java.lang.Object lower,java.lang.Object upper)` This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an unconstrained ASN.1 bit string value using the packed encoding rules (PER)into the stream.
void	`encode(Asn1PerOutputStream out,long lower,long upper)` This method encodes a size-constrained ASN.1 bit string value using the packed encoding rules (PER) into the stream.
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes ASN.1 bit string type using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes ASN.1 bit string type using the XML Encoding as specified in the XML schema standard.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix,java.lang.String[] namedbits,int[] namedbitindex)` This method encodes ASN.1 bit string type using the XML Encoding as specified in the XML schema standard.
void	`encodeAsString(Asn1JsonOutputStream outstream)` Encode ASN.1 BIT STRING to JSON.
void	`encodeContent(Asn1OerEncodeBuffer buffer)` This method encodes the content of an ASN.1 bit string value, according to OER.
void	`encodeV72(Asn1JsonOutputStream outstream)` Encode this bit string to JSON.
boolean	`equals(int nbits,byte[] value)` This method compares this bit string value to the given value for equality.
boolean	`equals(java.lang.Object bs_)` This method compares this bit string value to the given value for equality.

java.lang.String	getAsn1TypeName() This method gets the ASN.1 type name.
int	getLength() This method will return the length of the BIT STRING in bits.
int	getOerEffectiveMin() Return the lower bound for the OER effective size constraint.
int	hashCode() This method returns the hashCode for this bit string.
boolean	isNamedBitStr(java.lang.String buffer) This method determines is the input character string represented as named bit string or as bits sequence.
boolean	isSet(int bitno) This method tests if the given bit in the bit string is set or not.
void	set(int bitno) This method will set the given bit number to one (1).
void	set(int bitno,boolean bvalue) This method will set the given bit number to the given boolean value.
boolean[]	toBoolArray() This method converts the bit string stored in this object to a boolean array.
java.lang.String	toHexString() This method will return a hex string representation of the bit string value.
java.io.InputStream	toInputStream() This method will return a byte array input stream representation of the bit string value.
java.lang.String	toString() This method will return a string representation of the BIT STRING value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

(continued from last page)

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag TAG`

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 3).

HEX

`public static final short HEX`

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

BITS

`public static final short BITS`

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

ASN1VALUE

`public static final short ASN1VALUE`

The **ASN1VALUE**constant describes the string format for 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 String, else as Hex String.

HEXBIN

`public static final short HEXBIN`

The **HEXBIN**constant describes the string format for hex or binary digit value (e.g. 0xAA 10xxxxxx).

mStringFormat

`public static short mStringFormat`

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

numbits

`public transient int numbits`

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

value

`public transient byte value`

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.

trimZeroBits

`public boolean trimZeroBits`

This variable describes whether trailing zero bits should be truncated. This is required when encoding a named bit string using CER, DER, PER, or C-OER. By default, no trimming is done.

Constructors

(continued from last page)

Asn1BitString

`public` **`Asn1BitString`**`()`

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

Asn1BitString

`public` **`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

Asn1BitString

`public` **`Asn1BitString`**`(byte[] data)`

This constructor initializes a bit string with the given bytes, using all 8 bits of every byte.

Parameters:

`data` - Binary bit string contents

Asn1BitString

`public` **`Asn1BitString`**`(boolean[] 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

Asn1BitString

`public` **`Asn1BitString`**`(java.lang.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

Asn1BitString

`public` **`Asn1BitString`**`(java.util.BitSet bitSet)`

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

Parameters:

`bitSet` - Java BitSet object

Methods

getAsn1TypeName

`public java.lang.String` **`getAsn1TypeName`**`()`

(continued from last page)

This method gets the ASN.1 type name.

clear

```
public void clear(int bitno)
```

This method clears the given bit in the bit string.

Parameters:

`bitno` - Number of bit to clear. Bit numbers start at zero and with the MSB of the first byte and progress from left to right.

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of component or negative status value

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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

Throws:

`java.io.IOException`
`Asn1Exception`

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          long lower,
          java.lang.Object upper)
  throws Asn1Exception,
          java.io.IOException
```

This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER). This method is invoked when the MAX keyword is used as a boundary.

Parameters:

`buffer` - Decode message buffer object
`lower` - Lower bound (inclusive) of size constraint
`upper` - Upper bound (inclusive) of size constraint

Throws:

`java.io.IOException`
`Asn1Exception`

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          java.lang.Object lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER). This method is invoked when the MIN keyword is used as a boundary.

Parameters:

`buffer` - Decode message buffer object
`lower` - Lower bound (inclusive) of size constraint
`upper` - Upper bound (inclusive) of size constraint

Throws:

`java.io.IOException`
`Asn1Exception`

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          java.lang.Object lower,
          java.lang.Object upper)
  throws Asn1Exception,
          java.io.IOException
```

This method decodes a sized ASN.1 bit string value using the packed encoding rules (PER). This method is invoked when the MIN and MAX keywords are used as a boundary.

Parameters:

(continued from last page)

buffer - Decode message buffer object
lower - Lower bound (inclusive) of size constraint
upper - Upper bound (inclusive) of size constraint

Throws:

java.io.IOException
Asn1Exception

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variables within this class.

Parameters:

buffer - Encode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' 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

Throws:

java.io.IOException
Asn1Exception

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          java.lang.Object upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variables within this class. This method is called when the MAX keyword is used in code generation.

Parameters:

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

Throws:

java.io.IOException
Asn1Exception

(continued from last page)

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         java.lang.Object lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variables within this class. This method is called when the MIN keyword is used in code generation.

Parameters:

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

Throws:

`java.io.IOException`
`Asn1Exception`

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         java.lang.Object lower,
         java.lang.Object upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variables within this class. This method is called when both the MIN and MAX keywords are used in code generation.

Parameters:

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

Throws:

`java.io.IOException`
`Asn1Exception`

---

decode

```
public final void decode(Asn1NasDecodeBuffer buffer,
         int numbits)
  throws java.io.IOException
```

Decode the given number of bits from the buffer.

Parameters:

`buffer`
`numbits`

---

decodeRemainingBits

```
public final void decodeRemainingBits(Asn1NasDecodeBuffer buffer)
  throws java.io.IOException
```

Decode all remaining bits in the current container from the buffer. Containers are set by pushing/popping container lengths on/off the buffer's container stack. If no containers are on the stack, this decodes all remaining bits from the buffer.

Parameters:

`buffer`

decode

```
public void decode(Asn1MderDecodeBuffer buffer,
         int length)
  throws Asn1Exception,
         java.io.IOException
```

Decode a BIT STRING from the MDER encoding into this object. Exactly the given length of bits will be decoded.

Parameters:

`length` - This should be 8, 16, or 32, as these are the only lengths MDER supports. However, this is not checked here as it should be able to be checked at code generation time.

Throws:

`java.io.IOException`
`Asn1Exception`

decodeContent

```
public final void decodeContent(Asn1OerDecodeBuffer buffer,
         int numOctets,
         int unusedBits)
  throws java.io.IOException
```

This method decodes an ASN.1 BIT STRING's content that was encoded according to OER, given the total number of octets and the number of unused bits in the last octet.

Parameters:

`buffer` - Decode message buffer object
`numOctets` - Total number of octets encoding the content, including the final, partial octet (if unusedBits > 0)
`unusedBits` - # of unused bits in last octet.

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 BIT STRING that was encoded according to OER. This assumes the BIT STRING was not fixed-size constrained, so that the length and unused bits are in the encoding. Subclasses can override this method to simply call decodeContent for fixed-size constrained BIT STRINGs.

Parameters:

`buffer` - Decode message buffer object

encodeContent

```
public final void encodeContent(Asn1OerEncodeBuffer buffer)
```

This method encodes the content of an ASN.1 bit string value, according to OER. (The length determinant and # of unused bits is not encoded.)

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes this ASN.1 BIT STRING value, according to OER. This encodes the length determinant and # of unused bits, assuming that the BIT STRING is not fixed-size constrained. Subclasses may override this to simply call encodeContent for fixed-size constrained strings.

(continued from last page)

Parameters:

`buffer` - Encode message buffer object

---

getOerEffectiveMin

```
protected int getOerEffectiveMin()
```

Return the lower bound for the OER effective size constraint. When trimZeroBits is set, this controls the trimming done for OER encoding. Subclasses must override this to return the correct value if the BIT STRING has named bits and the effective size constraint has a lower bound other than zero.

---

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix,
         java.lang.String[] namedbits,
         int[] namedbitindex)
  throws java.io.IOException,
         Asn1Exception
```

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 name space prefix

`namedbits` - Array of named bits

`namedbitindex` - Arrat of named bits index values

Throws:

`java.io.IOException`

`Asn1Exception`

(continued from last page)

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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 prefix

Throws:

`java.io.IOException`
`Asn1Exception`

---

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 BIT STRING using the XML schema encoding rules.

Parameters:

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

---

isNamedBitStr

```
public boolean isNamedBitStr(java.lang.String buffer)
```

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 sequnece of identifiers.

---

decodeV72

```
public void decodeV72(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 bit string from JSON. This provides ObjSys-specific behavior that predates ITU-T X.697.

Parameters:

`buffer`

Throws:

`java.io.IOException`

---

encodeV72

```
public void encodeV72(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this bit string to JSON. This provides ObjSys-specific behavior that predates ITU-T X.697.

(continued from last page)

Parameters:

`outstream -` The output JSON stream.

Throws:

`java.io.IOException`

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 BIT STRING from JSON. This class's implementation decodes from a JSON object, as required by X.697 for a BIT STRING that is not constrained to a fixed length. Subclasses must override this for a BIT STRING that is constrained to a fixed length.

Parameters:

`buffer`

Throws:

`java.io.IOException`

decodeString

```
public final void decodeString(Asn1JsonDecodeBuffer buffer,
         int numbits)
  throws java.io.IOException
```

Decode ASN.1 BIT STRING from JSON. This function decodes from a JSON string, as required by X.697 for a BIT STRING that is constrained to a fixed length. This function ensures that the given number of bits are decoded and that unused bits are zero.

Parameters:

`buffer`

`numbits -` The constrained length.

Throws:

`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode ASN.1 BIT STRING to JSON. This class's implementation encodes to a JSON object, as required by X.697 for a BIT STRING that is not constrained to a fixed length. Subclasses must override this for a BIT STRING that is constrained to a fixed length.

Parameters:

`outstream -` The output JSON stream.

Throws:

`java.io.IOException`

encodeAsString

```
public final void encodeAsString(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode ASN.1 BIT STRING to JSON. This function encodes to a JSON string, as required by X.697 for a BIT STRING that is constrained to a fixed length.

Parameters:

(continued from last page)

`outstream` - The output JSON stream.

Throws:

`java.io.IOException`

equals

```
public boolean equals(int nbits,
                      byte[] 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:

`nbits` - Number of bits

`value` - Byte array containing bit data

equals

```
public boolean equals(java.lang.Object bs_)
```

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:

`bs_` - Bit string object to compare for equality

hashCode

```
public int hashCode()
```

This method returns the hashCode for this bit string.

getLength

```
public int getLength()
  throws Asn1InvalidLengthException
```

This method will return the length of the BIT STRING in bits.

Returns:

Number of bits.

isSet

```
public boolean isSet(int bitno)
```

This method tests if the given bit in the bit string is set or not.

Parameters:

`bitno` - Number of bit to set. Bit numbers start at zero and with the MSB of the first byte and progress from left to right.

Returns:

Bit value (true or false)

toBoolArray

```
public boolean[] toBoolArray()
```

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

Returns:

(continued from last page)

Boolean array value

---

set

```
public void set(int bitno,
         boolean bvalue)
```

This method will set the given bit number to the given boolean value. It will expand the existing bit array if it needs to.

Parameters:

`bitno` - Number of bit to set. Bit numbers start at zero and with the MSB of the first byte and progress from left to right.
`bvalue` - Bit value (true or false)

---

set

```
public void set(int bitno)
```

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

Parameters:

`bitno` - Number of bit to set. Bit numbers start at zero and with the MSB of the first byte and progress from left to right.

---

toInputStream

```
public java.io.InputStream toInputStream()
```

This method will return a byte array input stream representation of the bit string value.

Returns:

Reference to input stream object

---

toHexString

```
public java.lang.String toHexString()
```

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

---

toString

```
public java.lang.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:

Stringified representation of the value

---

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

(continued from last page)

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variables within this class.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variables within this class.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1MderOutputStream out,
          int length)
  throws Asn1Exception,
         java.io.IOException
```

Encode this BIT STRING into the MDER encoding. The length of this BIT STRING must match the given length.

Parameters:

`length` - This should be 8, 16, or 32, as these are the only lengths MDER supports. However, this is not checked here as it should be able to be checked at code generation time. We only check here that the actual and given lengths match.

com.objsys.asn1j.runtime
Class Asn1BMPString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1BMPString
```

public class **Asn1BMPString**
extends Asn1CharString

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

Field Summary

static final int	BITSPERCHAR The **BITSPERCHAR**constant specifies the number of bits per character for PER (aligned or unaligned). Value: **16**
static final Asn1CharSet	CHARSET
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 30).

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1BMPString() The default constructor creates an empty string object.
public	Asn1BMPString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 BMP string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode the value in accordance with OER.
void	`decode(Asn1OerDecodeBuffer buffer,int length)` Decode the value in accordance with OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,Asn1CharSet charSet)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1OerEncodeBuffer buffer)` Encode the value in accordance with OER.
void	`encode(Asn1OerEncodeBuffer buffer,boolean withLength)` Encode the string, with or without a length determinant.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,Asn1CharSet charSet)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 BMP string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,Asn1CharSet charSet)` This method encodes an ASN.1 BMP string value in accordance with the packed encoding rules (PER).

void	encode(Asn1PerOutputStream out,Asn1CharSet charSet,long lower,long upper) This overloaded version of the encode method encodes an ASN.1 BMP string value in accordance with the packed encoding rules (PER).

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

CHARSET

public static final com.objsys.asn1j.runtime.Asn1CharSet **CHARSET**

BITSPERCHAR

public static final int **BITSPERCHAR**

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

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 30).

Constructors

Asn1BMPString

public **Asn1BMPString**()

The default constructor creates an empty string object.

Asn1BMPString

public **Asn1BMPString**(java.lang.String data)

(continued from last page)

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

(continued from last page)

`charSet` - Object representing permitted alphabet constraint character set (optional)

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         Asn1CharSet charSet,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object
`charSet` - Object representing permitted alphabet constraint character set (optional)

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         Asn1CharSet charSet,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:

`buffer` - Encode message buffer object
`charSet` - Object representing the permitted alphabet constraint character set

(continued from last page)

`lower` - Effective size constraint lower bound
`upper` - Effective size constraint upper bound

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode the value in accordance with OER. This class's implementation decodes the string with a length determinant. If a subclass should be decoded without a length determinant, it should override this method to invoke decode(buffer, length). Subclasses may override this to add constraint checks after invoking this method to decode the string.

Parameters:

`buffer`

---

decode

```
public final void decode(Asn1OerDecodeBuffer buffer,
         int length)
  throws java.io.IOException
```

Decode the value in accordance with OER. This class's implementation decodes a string of the given length. This method is final as I don't see any reason for overriding it.

Parameters:

`buffer`
`length` - Length of string to decode, in characters.

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode the value in accordance with OER. This class's implementation invokes encode(buffer, true) to encode the string with a length determinant. If a subclass should be encoded without a length determinant, it should override this to invoke encode(buffer, false).

Parameters:

`buffer`

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer,
         boolean withLength)
  throws java.io.IOException
```

Encode the string, with or without a length determinant. Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters:

`buffer`
`withLength` - true if a length determinant should be encoded.

---

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

(continued from last page)

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:
`out` - PER Output Stream object

Throws:
`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
          Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:
`out` - PER Output Stream object
`charSet` - Object representing permitted alphabet constraint character set (optional)

Throws:
`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:
`out` - 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

Throws:

(continued from last page)

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1Boolean

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Boolean
```

public class **Asn1Boolean**
extends Asn1Type

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

Field Summary

static final Asn1Boolean	FALSE_VALUE The **FALSE_VALUE**constant represents a boolean FALSE value.
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 1).
static final Asn1Boolean	TRUE_VALUE The **TRUE_VALUE**constant represents a boolean TRUE value.
boolean	value This public member variable is where the boolean value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Boolean() The default constructor sets the boolean value to false.
public	Asn1Boolean(boolean value_) This constructor creates a boolean object from a boolean value.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 boolean value including the UNIVERSAL tag value and length if explicit tagging is specified.

void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 BOOLEAN from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode BOOLEAN value according to OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 boolean value using the Packed Encoding Rules (PER).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 boolean value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 boolean value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 boolean value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this boolean value to JSON.
void	`encode(Asn1OerEncodeBuffer buffer)` Encode BOOLEAN value according to OER.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 boolean value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 boolean value using the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncodeBuffer buffer)` This method encodes an ASN.1 boolean value using the XML encoding rules (XER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 boolean value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 boolean value according to the Obj-Sys XML encoding rules.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix,boolean asText)` This method encodes an ASN.1 boolean value.
void	`encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName)` This method encodes an ASN.1 boolean value using the XML Encoding as specified in the W3C XML schema standard(asn2xsd).

boolean	equals(boolean bvalue) This method compares this boolean value to the given value for equality.
boolean	equals(java.lang.Object bvalue) This method compares this boolean value to the given value for equality.
java.lang.String	getAsn1TypeName() Returns the ASN.1 type name.
int	hashCode() This method returns the hashCode of the current object.
static void	setTrueEncodedByte(byte b) This method sets the byte value that represents the boolean value TRUE.
java.lang.String	toString() This method will return a string representation of the boolean value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 1).

TRUE_VALUE

public static final com.objsys.asn1j.runtime.Asn1Boolean **TRUE_VALUE**

The **TRUE_VALUE**constant represents a boolean TRUE value.

FALSE_VALUE

public static final com.objsys.asn1j.runtime.Asn1Boolean **FALSE_VALUE**

The **FALSE_VALUE**constant represents a boolean FALSE value.

value

public transient boolean **value**

(continued from last page)

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.

Constructors

Asn1Boolean

```
public Asn1Boolean()
```

The default constructor sets the boolean value to false.

Asn1Boolean

```
public Asn1Boolean(boolean value_)
```

This constructor creates a boolean object from a boolean value.

Parameters:

`value_` - Boolean value

Methods

getAsn1TypeName

```
public java.lang.String getAsn1TypeName()
```

Returns the ASN.1 type name.

Returns:

The ASN.1 type name BOOLEAN.

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**in this object.

Parameters:

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

setTrueEncodedByte

```
public static void setTrueEncodedByte(byte b)
```

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, a valid representation in BER, CER, and DER.

Parameters:

`b` - The byte value to set.

(continued from last page)

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length (in octets) of encoded component

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - PER Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - PER Encode message buffer object

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode BOOLEAN value according to OER.

Parameters:

`buffer`

Throws:

`IOException`

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode BOOLEAN value according to OER.

Parameters:

`buffer`

(continued from last page)

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object
`elemName` - Element name

---

encode

```
public void encode(Asn1XerEncodeBuffer buffer)
  throws Asn1Exception
```

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

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 boolean value according to the Obj-Sys XML encoding rules. It encodes the value as XML text.

Parameters:

`buffer` - Encode message buffer object
`elemName` - Element name for optional surrounding element.
`nsPrefix` - XML element name space prefix

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix,
         boolean asText)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 boolean value. It is for use with extended-XER.

Parameters:

(continued from last page)

`buffer` - Encode message buffer object
`elemName` - Element name for optional surrounding element.
`nsPrefix` - XML element name space prefix
`asText` - If true, encode as text. Otherwise, encode as an empty element.

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws java.io.IOException,
         Asn1Exception
```

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` - Attribute name

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 boolean value using the XML schema encoding rules.

Parameters:

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

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 BOOLEAN from JSON.

Parameters:

`buffer`

Throws:

`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this boolean value to JSON.

equals

```
public boolean equals(boolean bvalue)
```

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

Parameters:

`bvalue` - Boolean test value (boolean value)

(continued from last page)

equals

```
public boolean equals(java.lang.Object bvalue)
```

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

Parameters:

`bvalue` - Boolean test value (Asn1Boolean object)

hashCode

```
public int hashCode()
```

This method returns the hashCode of the current object.

toString

```
public java.lang.String toString()
```

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

Returns:

Stringified representation of the value

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - PER Encode message stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1CanonicalException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1CanonicalException
```

public class **Asn1CanonicalException**
extends Asn1Exception

This exception is thrown when an encoding violates the canonical encoding rules for the current encoding rule set.

Constructor Summary

public	Asn1CanonicalException(java.lang.String msg)

Methods inherited from class java.lang.Throwable
addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1CanonicalException

public **Asn1CanonicalException**(java.lang.String msg)

com.objsys.asn1j.runtime
Class Asn1CerInputStream

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
               ┗-com.objsys.asn1j.runtime.Asn1BerInputStream
                  ┗-com.objsys.asn1j.runtime.Asn1CerInputStream
```

public class **Asn1CerInputStream**
extends Asn1BerInputStream

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

Fields inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer
mByteCount

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1CerInputStream(java.io.InputStream istream) This constructor creates a CER input stream object that references an encoded ASN.1 message.

Methods inherited from class com.objsys.asn1j.runtime.Asn1BerInputStream
available, close, mark, markSupported, reset, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
calcIndefLen, decodeEnumValue, decodeEnumValue, decodeLength, decodeOpenType, decodeOpenType, decodeTag, decodeTagAndLength, getLastTag, matchTag, matchTag, matchTag, movePastEOC, parse, peekTag, peekTag, readByte

Methods inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer
addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1CerInputStream

public **Asn1CerInputStream**(java.io.InputStream istream)

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

Parameters:

istream - Input stream containing an encoded ASN.1 message.

com.objsys.asn1j.runtime
Class Asn1CerOutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-com.objsys.asn1j.runtime.Asn1OutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1BerOutputStream
            |
            +-com.objsys.asn1j.runtime.Asn1CerOutputStream
```

public class **Asn1CerOutputStream**
extends Asn1BerOutputStream

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OutputStream`
`os`

Constructor Summary

`public`	`Asn1CerOutputStream(java.io.OutputStream os)` This constructor creates a CER output stream object with default size of buffer.
`public`	`Asn1CerOutputStream(java.io.OutputStream os,int bufSize)` This constructor creates a buffered CER output stream object.

Method Summary

`void`	`encode(Asn1Type object,boolean explicit)` This method encodes and writes to the stream ASN.1 types.
`void`	`encodeBitString(byte[] value,int numbits,boolean explicit,Asn1Tag tag)` This method writes the given array of bytes as bit string value.
`void`	`encodeBMPString(java.lang.String value,boolean explicit,Asn1Tag tag)` This method writes the given string as BMP string value.
`void`	`encodeCharString(java.lang.String value,boolean explicit,Asn1Tag tag)` This method encodes and writes to the stream an ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
`void`	`encodeOctetString(byte[] value,boolean explicit,Asn1Tag tag)` This method writes the given array of bytes as octet string value.

void	encodeStringTag(int nbytes,Asn1Tag tag) This method encodes and writes both a tag and length value to the stream.
void	encodeStringTag(int nbytes,short tagClass,short tagForm,int tagIDCode) This method encodes and writes both a tag and length value to the stream.
void	encodeUnivString(int[] value,boolean explicit,Asn1Tag tag) This method writes the given array of integers as UniversalString value.
boolean	isBER() This method is used for testing encode rules type.

Methods inherited from class com.objsys.asn1j.runtime.Asn1BerOutputStream

encode, encodeBitString, encodeBMPString, encodeCharString, encodeEOC, encodeIdentifier, encodeIntValue, encodeLength, encodeOctetString, encodeTag, encodeTag, encodeTagAndIndefLen, encodeTagAndIndefLen, encodeTagAndLength, encodeUnivString, isBER

Methods inherited from class com.objsys.asn1j.runtime.Asn1OutputStream

close, flush, getContext, write, write, write, write2Bytes, write4Bytes

Methods inherited from class java.io.OutputStream

close, flush, write, write, write

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1CerOutputStream

public **Asn1CerOutputStream**(java.io.OutputStream os)

This constructor creates a CER output stream object with default size of buffer. Whenever the buffer becomes full, the buffer will be flushed to the stream. Note: in Java ME environment, the stream is not buffered.

Parameters:

os - The underlying OutputStream object.

Asn1CerOutputStream

public **Asn1CerOutputStream**(java.io.OutputStream os,
int bufSize)

This constructor creates a buffered CER output stream object. Whenever the buffer becomes full, the buffer will be flushed to the stream. Note: in Java ME environment, the stream is not buffered.

Parameters:

os - The underlying OutputStream object.

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

Methods

encodeBMPString

```
public void encodeBMPString(java.lang.String value,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

This method writes the given string as BMP string value.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeBitString

```
public void encodeBitString(byte[] value,
         int numbits,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeCharString

```
public void encodeCharString(java.lang.String value,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

encodeOctetString

```
public void encodeOctetString(byte[] value,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encodeStringTag

```
public void encodeStringTag(int nbytes,
         Asn1Tag tag)
  throws java.io.IOException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeStringTag

```
public void encodeStringTag(int nbytes,
         short tagClass,
         short tagForm,
         int tagIDCode)
  throws java.io.IOException
```

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

Parameters:

`nbytes` - The number of bytes in string to be encoded.
`tagClass` - The class of the tag to be encoded.
`tagForm` - The form of the tag to be encoded.
`tagIDCode` - The ID code of the tag to be encoded.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeUnivString

```
public void encodeUnivString(int[] value,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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

Parameters:

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

Throws:

`IOException -` Any exception thrown by the underlying OutputStream.
`Asn1Exception -` Thrown, if operation is failed.

---

encode

```
public void encode(Asn1Type object,
         boolean explicit)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`object -` The object to be written
`explicit -` Flag indicating explicit tagging should be done

Throws:

`IOException -` Any exception thrown by the underlying OutputStream.

---

isBER

```
public boolean isBER()
```

This method is used for testing encode rules type.

com.objsys.asn1j.runtime
Class Asn1CharOutputStream

```
java.lang.Object
   |
   +-java.io.Writer
      |
      +-com.objsys.asn1j.runtime.Asn1CharOutputStream
```

Direct Known Subclasses:
Asn1JsonOutputStream

---

public class **Asn1CharOutputStream**
extends java.io.Writer

Base class for character-based output. This class also aides in managing indentation. By default, the indent() method will output whitespace, but this can be disabled using setWriteWhitespace. This class is preferable to Asn1OutputStream when character data is being written.

Fields inherited from class `java.io.Writer`
`lock`

Constructor Summary

public	`Asn1CharOutputStream(java.io.Writer writer)` Create character output stream on the given Writer.

Method Summary

void	`close()`
void	`decrLevel()` This method decrements the indentation level.
void	`flush()`
void	`incrLevel()` This method increments the indentation level.
void	`indent()` This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the output stream.
void	`setWriteWhitespace(boolean value)` Turn whitespace writing by the indent() method on/off.
void	`write(char[] cbuf,int off,int len)`

Methods inherited from class `java.io.Writer`

append, append, append, close, flush, write, write, write, write, write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1CharOutputStream

`public` **`Asn1CharOutputStream`**`(java.io.Writer writer)`

Create character output stream on the given Writer. This constructor allows you to send output to any Writer at all.

Methods

decrLevel

`public void` **`decrLevel`**`()`

This method decrements the indentation level.

incrLevel

`public void` **`incrLevel`**`()`

This method increments the indentation level.

indent

```
public void indent()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

setWriteWhitespace

`public void` **`setWriteWhitespace`**`(boolean value)`

Turn whitespace writing by the indent() method on/off.

Parameters:

`value`

close

```
public void close()
  throws java.io.IOException
```

flush

```
public void flush()
  throws java.io.IOException
```

write

```
public void write(char[] cbuf,
         int off,
         int len)
  throws java.io.IOException
```

com.objsys.asn1j.runtime
Class Asn1CharRange

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1CharSet
      |
      +-com.objsys.asn1j.runtime.Asn1CharRange
```

public class **Asn1CharRange**
extends Asn1CharSet

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 charater string components containing permitted alphabet constraints.

Field Summary

int	`mLower` This variable represents the lower value of the range.
int	`mUpper` This variable represents the upper value of the range.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1CharSet`
`mABitsPerChar`, `mUBitsPerChar`

Constructor Summary

public	`Asn1CharRange(int lower,int upper)` This constructor sets the range values.

Method Summary

int	`getCharAtIndex(int index)` This method will fetch the character from the permitted alphabet at the given index.
int	`getCharIndex(int charValue)` This method will determine the index of the given character within the permitted alphabet character set.

int	`getMaxValue()` This method will determine the maximum value of the given character within the permitted alphabet character set.
boolean	`validate(java.lang.String s)` This method will validate a character string by comparing its contents to the character range.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharSet`
`getCharAtIndex, getCharIndex, getMaxValue, getNumBitsPerChar, validate`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

mLower

`protected int` **`mLower`**

This variable represents the lower value of the range.

mUpper

`protected int` **`mUpper`**

This variable represents the upper value of the range.

Constructors

Asn1CharRange

`public` **`Asn1CharRange`**`(int lower, int upper)`

This constructor sets the range values.

Parameters:

`lower` - Range lower value
`upper` - Range upper value

Methods

getCharAtIndex

`public int` **`getCharAtIndex`**`(int index) throws Asn1ConsVioException`

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

Parameters:

`index` - Index of character within the character set

(continued from last page)

Returns:
Character at given index

Throws:
`Asn1ConsVioException` - Index not within define range

---

getCharIndex

```
public int getCharIndex(int charValue)
  throws Asn1ConsVioException
```

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

Throws:
`Asn1ConsVioException` - Character not found in set

---

getMaxValue

```
public int getMaxValue()
```

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

Returns:
Upper Bound Character or Character with max int value

---

validate

```
public boolean validate(java.lang.String s)
```

This method will validate a character string by comparing its contents to the character range. Each character in the string is checked against the range. If it exceeds the upper or lower limit of the range, false is returned. Otherwise true is returned.

Parameters:
`s` - The string to be validated.

Returns:
False if the string contains invalid characters; true otherwise.

com.objsys.asn1j.runtime
Class Asn1CharSet

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1CharSet
```

Direct Known Subclasses:
Asn1DiscreteCharSet, Asn1CharRange

---

public abstract class **Asn1CharSet**
extends java.lang.Object

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

Field Summary

int	`mABitsPerChar` This variable holds number of bits-per-character (PER aligned).
int	`mUBitsPerChar` This variable holds number of bits-per-character (PER unaligned).

Constructor Summary

protected	`Asn1CharSet(int nchars)` This constructor sets the number of bits-per-character values based on the given number of characters in the character set.

Method Summary

abstract int	`getCharAtIndex(int index)` This method will fetch the character from the permitted alphabet at the given index.
abstract int	`getCharIndex(int charValue)` This method will determine the index of the given character within the permitted alphabet character set.
abstract int	`getMaxValue()` This method will determine the maximum value of the given character within the permitted alphabet character set.
int	`getNumBitsPerChar(boolean aligned)` This method will return the number of bits-per-character.
abstract boolean	`validate(java.lang.String s)` This method will validate a character string by comparing its contents to the character set.

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mABitsPerChar

protected int **mABitsPerChar**

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

mUBitsPerChar

protected int **mUBitsPerChar**

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

Constructors

Asn1CharSet

protected **Asn1CharSet**(int nchars)

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

Methods

getCharAtIndex

public abstract int **getCharAtIndex**(int index)
  throws Asn1ConsVioException

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

Throws:

Asn1ConsVioException - Index not within define range

getCharIndex

public abstract int **getCharIndex**(int charValue)
  throws Asn1ConsVioException

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

Parameters:

charValue - Character value to search for

Returns:

(continued from last page)

Index of character

Throws:

`Asn1ConsVioException` - Character not found in set

getMaxValue

`public abstract int` **`getMaxValue`**`()`

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

Returns:

Upper Bound Character or Character with max int value

getNumBitsPerChar

`public int` **`getNumBitsPerChar`**`(boolean aligned)`

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.

validate

`public abstract boolean` **`validate`**`(java.lang.String s)`

This method will validate a character string by comparing its contents to the character set. If a character string contains characters that are not in the character set, this method will return false. Otherwise it returns true.

Parameters:

`s` - The string to be validated.

Returns:

False if the string contains invalid characters; true otherwise.

com.objsys.asn1j.runtime
Class Asn1CharString

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Type
      |
      +-com.objsys.asn1j.runtime.Asn1CharString
```

Direct Known Subclasses:
Asn1VarWidthCharString, Asn1UTF8String, Asn1OID_IRI, Asn1BMPString, Asn18BitCharString

---

public abstract class **Asn1CharString**
extends Asn1Type

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.

Field Summary

java.lang.StringBuffer	mStringBuffer The **mStringBuffer**member variable is used to do internal operations on a string being encoded or decoded.
java.lang.String	value The **value**public member variable is used to hold the string value to be encoded or the results of a decode operation.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

protected	Asn1CharString(short typeCode) This constructor creates an empty string that can be used in a decode method call to receive a string value.
protected	Asn1CharString(java.lang.String data,short typeCode) This constructor initializes a character string from the given string data.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength,Asn1Tag tag) This method decodes an ASN.1 character string value including the UNIVERSAL tag value and length if explicit tagging is specified.

void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 restricted character string from JSON.
void	`decode(Asn1PerDecodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decodeByteToChar(Asn1OerDecodeBuffer buffer,int length)` This method decodes a given number of bytes and converts each byte to a char having the same value ("same" value if you treat the bytes as being unsigned).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit,Asn1Tag tag)` This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
static int	`encode(Asn1BerEncodeBuffer buffer,java.lang.String value,boolean explicit,Asn1Tag tag)` Encode the given string value, as a character string, according to BER, encoding a single byte per character.
void	`encode(Asn1JsonOutputStream out)` Encode the value of this object as a JSON string.
void	`encode(Asn1PerEncodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes ASN.1 8-bit character string types including IA5String, PrintableString, NumericString, etc.
boolean	`equals(java.lang.Object cs)` This method compares this character string value to the given value for equality.

boolean	`equals(java.lang.String value)` This method compares this character string value to the given value for equality.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
int	`getLength()` This method will return the length of the character string in characters.
int	`hashCode()` This method returns the hashCode for this character string.
java.lang.String	`toString()` This method will return a string representation of the value.
boolean	`validate(Asn1CharSet charSet)` This method will attempt to validate a string against its internal character set.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

value

`public transient java.lang.String` **`value`**

The **value**public member variable is used to hold the string value to be encoded or the results of a decode operation.

mStringBuffer

`protected transient java.lang.StringBuffer` **`mStringBuffer`**

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

Constructors

Asn1CharString

`protected` **`Asn1CharString`**`(short typeCode)`

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

Parameters:

(continued from last page)

`typeCode` - Universal ID code for ASN.1 character string

Asn1CharString

```
protected Asn1CharString(java.lang.String data,
                         short typeCode)
```

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

Parameters:

`data` - Character string
`typeCode` - Universal ID code for ASN.1 character string

Methods

getAsn1TypeName

```
public java.lang.String getAsn1TypeName()
```

Returns the ASN.1 type name.

Returns:

The ASN.1 type name based on the character string typecode.

---

decode

```
protected void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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. Since character strings in Java are encoded in two bytes by default, this method explicitly takes the lower byte from the ASN.1-byte to Java-character conversion.

Parameters:

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

---

encode

```
protected static int encode(Asn1BerEncodeBuffer buffer,
         java.lang.String value,
         boolean explicit,
         Asn1Tag tag)
```

Encode the given string value, as a character string, according to BER, encoding a single byte per character. If explicit tagging is specified, the given tag value and a length are also encoded.

Parameters:

`buffer`
`explicit`
`tag`
`value`

Returns:

encode

```
protected int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit,
         Asn1Tag tag)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done
`tag` - Universal tag to apply

Returns:

Length in octets of encoded component

decode

```
protected void decode(Asn1PerDecodeBuffer buffer,
         int abpc,
         int ubpc,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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)

decode

```
protected void decode(Asn1PerDecodeBuffer buffer,
         int abpc,
         int ubpc,
         Asn1CharSet charSet,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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

(continued from last page)

encode

```
protected void encode(Asn1PerEncodeBuffer buffer,
          int abpc,
          int ubpc,
          Asn1CharSet charSet)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**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)

---

encode

```
protected void encode(Asn1PerEncodeBuffer buffer,
          int abpc,
          int ubpc,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**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

---

decodeByteToChar

```
public final void decodeByteToChar(Asn1OerDecodeBuffer buffer,
          int length)
  throws java.io.IOException
```

This method decodes a given number of bytes and converts each byte to a char having the same value ("same" value if you treat the bytes as being unsigned).

Parameters:

`buffer`
`length` - Length of string to decode

---

encode

```
public void encode(Asn1XerEncoder buffer,
          java.lang.String elemName)
  throws java.io.IOException,
          Asn1Exception
```

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

(continued from last page)

Parameters:

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

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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 name space prefix

---

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

Parameters:

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

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 restricted character string from JSON.

Parameters:

`buffer`

Throws:

`java.io.IOException`

(continued from last page)

encode

```
public void encode(Asn1JsonOutputStream out)
  throws java.io.IOException
```

Encode the value of this object as a JSON string.

Parameters:

`out`

---

equals

```
public boolean equals(java.lang.String value)
```

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

Parameters:

`value` - String value

---

hashCode

```
public int hashCode()
```

This method returns the hashCode for this character string.

---

equals

```
public boolean equals(java.lang.Object cs)
```

This method compares this character string value to the given value for equality. Strings that are orthographically identical but of different types (e.g., UTF8String "123" and NumericString "123") are not equal.

Parameters:

`cs` - Character string object reference

---

getLength

```
public int getLength()
  throws Asn1InvalidLengthException
```

This method will return the length of the character string in characters.

Returns:

Number of characters.

---

validate

```
public boolean validate(Asn1CharSet charSet)
```

This method will attempt to validate a string against its internal character set.

Returns:

True or False.

---

toString

```
public java.lang.String toString()
```

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

Returns:

Stringified representation of the value

com.objsys.asn1j.runtime
Class Asn1Choice

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Choice
```

public abstract class **Asn1Choice**
extends Asn1Type

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

Field Summary

int	choiceID This member variable is where the selected choice option identifier is stored.
Asn1Type	element This member variable is where the selected choice option value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Choice() The default constructor initializes the choiceID and value.

Method Summary

java.lang.Object	clone() Return a clone of this choice.
boolean	equals(java.lang.Object cv_) This method compares this type element with the passed type element.
java.lang.String	getAsn1TypeName() Returns the ASN.1 type name.
int	getChoiceID() This method returns the choice identifier.

Asn1Type	getElement() This method returns the element object.
abstract java.lang.String	getElemName() This abstract method return the name of the selected element.
int	hashCode() This method returns the hashcode for the chosen element.
void	setElement(int choiceID,Asn1Type element) This protected method sets the choice ID and value to the given values.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

choiceID

protected transient int **choiceID**

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.

element

protected transient com.objsys.asn1j.runtime.Asn1Type **element**

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.

Constructors

Asn1Choice

public **Asn1Choice**()

The default constructor initializes the choiceID and value.

Methods

getAsn1TypeName

public java.lang.String **getAsn1TypeName**()

(continued from last page)

Returns the ASN.1 type name.

Returns:

The ASN.1 type name CHOICE.

getChoiceID

```
public int getChoiceID()
```

This method returns the choice identifier.

getElement

```
public Asn1Type getElement()
```

This method returns the element object.

getElemName

```
public abstract java.lang.String getElemName()
```

This abstract method return the name of the selected element. A concrete version is generated by the compiler.

setElement

```
public void setElement(int choiceID,
         Asn1Type element)
```

This protected method sets the choice ID and value to the given values. It can only be invoked by compiler-generated set_<element> methods.

equals

```
public boolean equals(java.lang.Object cv_)
```

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

Parameters:

`cv_` - Asn1Choice type value

hashCode

```
public int hashCode()
```

This method returns the hashcode for the chosen element. If the element is null, the hashCode returned is 1, to avoid changing any other hash codes that depend on it.

clone

```
public java.lang.Object clone()
  throws java.lang.CloneNotSupportedException
```

Return a clone of this choice. The clone is returned as a memberwise clone.

com.objsys.asn1j.runtime
Class Asn1ChoiceExt

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
         ┗-com.objsys.asn1j.runtime.Asn1OpenType
            ┗-com.objsys.asn1j.runtime.Asn1ChoiceExt
```

public class **Asn1ChoiceExt**
extends Asn1OpenType

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).

Field Summary

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.
Asn1Tag	tag tag is used with OER.

Fields inherited from class com.objsys.asn1j.runtime.Asn1OpenType
BER, dataEncoding, EDATAMSG, JSON, mEncodeBuffer, mLength, OER, PER, UNKNOWN, XER

Fields inherited from class com.objsys.asn1j.runtime.Asn1OctetString
TAG, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1ChoiceExt()

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an extension field using the Basic Encoding Rules (BER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an open type extension in a CHOICE construct using the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).
void	`encode(Asn1OerEncodeBuffer buffer)` Encode this unknown choice extension using the Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER).

Methods inherited from class `com.objsys.asn1j.runtime.Asn1OpenType`

`decode, decode, decode, decode, decode72, decodeExtension, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode72, encodeAsExtension, encodeAsExtension, encodeAsExtension, getAsn1TypeName, getCharData, getDataEncoding, getSaxHandler, getSaxHandler, setBinaryData, setCharData, setCharData, toString`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1OctetString`

`compareTo, decode, decode, decode, decode, decode, decode, decode, decode, decodeAsBase64, decodeAsHex, decodeContent, decodeRemainingBits, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsBase64, encodeAsHex, encodeAttribute, encodeBase64Binary, encodeContent, equals, equals, equals, getAsn1TypeName, getLength, getMderLength, hashCode, toInputStream, toString`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

(continued from last page)

Fields

choiceIndex

```
public 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.

tag

```
public com.objsys.asn1j.runtime.Asn1Tag tag
```

tag is used with OER. When decoding, it should be set to hold the decoded tag. When encoding, it is encoded as the tag for the unknown choice extension. The form (tag.mForm) is irrelevant.

Constructors

Asn1ChoiceExt

```
public Asn1ChoiceExt()
```

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - Decode message buffer object

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`explicit` - Flag indicating element is explicitly tagged

Returns:

Length of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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 `value`public member list variable. The public member variable `choiceIndex`will be populated with the decoded choice index value.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1BerOutputStream out,
          boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating element is explicitly tagged

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - PER Output Stream object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode this unknown choice extension using the Octet Encoding Rules (OER). This uses this object's tag as the tag, and then encodes the value field as open type content.

com.objsys.asn1j.runtime
Class Asn1ConsVioException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1ConsVioException
```

public class **Asn1ConsVioException**
extends Asn1Exception

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..

Constructor Summary

public	`Asn1ConsVioException(java.lang.String varname,long value)` This constructor creates an exception object with a standard message based on the given variable name and value.
public	`Asn1ConsVioException(java.lang.String varname,double value)` This constructor creates an exception object with a standard message based on the given variable name and value.
public	`Asn1ConsVioException(java.lang.String varname,java.lang.String value)` This constructor creates an exception object with a standard message based on the given variable name and value.

Methods inherited from class `java.lang.Throwable`
`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1ConsVioException

```
public Asn1ConsVioException(java.lang.String varname,
                            long value)
```

(continued from last page)

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

Asn1ConsVioException

```
public Asn1ConsVioException(java.lang.String varname,
                            double 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

Asn1ConsVioException

```
public Asn1ConsVioException(java.lang.String varname,
                            java.lang.String 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

com.objsys.asn1j.runtime
Class Asn1Context

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Context
```

public class **Asn1Context**
extends java.lang.Object

This class is a holder for things which are to be contained in an encoder or decoder. Since we have multiple base classes for encoders/decoders (Asn1MessageBuffer and Asn1OutputStream), this class provides a single container for all of those things which would otherwise be contained in the various base classes. This class also provides a location for some convenience methods related to the contained objects (e.g. getCurrentElement). Thread-safety: a given instance of this class should be accessed by a single thread.

Field Summary

Asn1NamedEventDispatcher	**eventDispatcher** The named event dispatcher for this context.

Constructor Summary

public	**Asn1Context()**

Method Summary

void	**enableElementTracking()** Enable element name tracking.
java.lang.String	**getCurrentElement()** Return the name of the current element.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

eventDispatcher

`public com.objsys.asn1j.runtime.Asn1NamedEventDispatcher` **`eventDispatcher`**

The named event dispatcher for this context.

Constructors

(continued from last page)

Asn1Context

`public` **`Asn1Context`**`()`

Methods

enableElementTracking

`public void` **`enableElementTracking`**`()`

Enable element name tracking. This can be useful for error reporting. Element name tracking must be enabled for getCurrentElement to be of any use.

---

getCurrentElement

`public java.lang.String` **`getCurrentElement`**`()`

Return the name of the current element. This is only useful if you have called enableElementTracking at the start of encoding/decoding. The name returned will be in the form element1.element2..., If you have not enabled element tracking, an exception will be thrown.

com.objsys.asn1j.runtime
Class Asn1Date

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1Date
```

public class **Asn1Date**
extends Asn18BitCharString

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 31).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString

BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Date() The default constructor creates an empty string object.
public	Asn1Date(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 DATE string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	decode(Asn1OerDecodeBuffer buffer) This method decodes an ASN.1 DATE value in accordance with the octet encoding rules (OER).
void	decode(Asn1PerDecodeBuffer buffer) This method decodes an ASN.1 DATE value in accordance with the packed encoding rules (PER).
int	encode(Asn1BerEncodeBuffer buffer,boolean explicit) This method encodes an ASN.1 DATE string type.
void	encode(Asn1BerOutputStream out,boolean explicit) This method encodes and writes to the stream an ASN.1 DATE value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	encode(Asn1OerEncodeBuffer buffer) This method encodes an ASN.1 DATE value in accordance with the octet encoding rules (OER).
void	encode(Asn1PerEncodeBuffer buffer) This method encodes an ASN.1 DATE value in accordance with the packed encoding rules (PER).
void	encode(Asn1PerOutputStream out) This method encodes an ASN.1 DATE value in accordance with the packed encoding rules (PER) directly into the stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 31).

Constructors

Asn1Date

```
public Asn1Date()
```

The default constructor creates an empty string object.

Asn1Date

```
public Asn1Date(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
                   java.io.IOException
```

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

Parameters:

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

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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

Parameters:

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

Returns:

Length in octets of encoded component

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`java.io.IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 DATE value in accordance with the octet encoding rules (OER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 DATE value in accordance with the octet encoding rules (OER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 DATE value in accordance with the packed encoding rules (PER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 DATE value in accordance with the packed encoding rules (PER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 DATE 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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object

Throws:

`java.io.IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1DateTime

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1DateTime
```

public class **Asn1DateTime**
extends Asn18BitCharString

This is a container class for holding the components of an ASN.1 DATE-TIME value.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 33).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1DateTime() The default constructor creates an empty string object.
public	Asn1DateTime(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 DATE-TIME string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	decode(Asn1OerDecodeBuffer buffer) This method decodes an ASN.1 DATE-TIME value in accordance with the octet encoding rules (OER).
void	decode(Asn1PerDecodeBuffer buffer) This method decodes an ASN.1 DATE-TIME value in accordance with the packed encoding rules (PER).
int	encode(Asn1BerEncodeBuffer buffer,boolean explicit) This method encodes an ASN.1 DATE-TIME string type.
void	encode(Asn1BerOutputStream out,boolean explicit) This method encodes and writes to the stream an ASN.1 DATE-TIME value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	encode(Asn1OerEncodeBuffer buffer) This method encodes an ASN.1 DATE-TIME value in accordance with the octet encoding rules (OER).
void	encode(Asn1PerEncodeBuffer buffer) This method encodes an ASN.1 DATE-TIME value in accordance with the packed encoding rules (PER).
void	encode(Asn1PerOutputStream out) This method encodes an ASN.1 DATE-TIME value in accordance with the packed encoding rules (PER) directly into the stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn18BitCharString

decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString

decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 33).

Constructors

Asn1DateTime

```
public Asn1DateTime()
```

The default constructor creates an empty string object.

Asn1DateTime

```
public Asn1DateTime(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
                   java.io.IOException
```

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

Parameters:

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

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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

Parameters:

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

Returns:

Length in octets of encoded component

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`java.io.IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 DATE-TIME value in accordance with the octet encoding rules (OER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 DATE-TIME value in accordance with the octet encoding rules (OER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 DATE-TIME value in accordance with the packed encoding rules (PER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 DATE-TIME value in accordance with the packed encoding rules (PER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 DATE-TIME 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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object

Throws:

`java.io.IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1DecodeBitBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1DecodeBitBuffer
```

All Implemented Interfaces:
Asn1BitMessageBuffer

Direct Known Subclasses:
Asn1PerDecodeBuffer, Asn1OerDecodeBuffer, Asn1NasDecodeBuffer

---

public class **Asn1DecodeBitBuffer**
extends Asn1DecodeBuffer
implements Asn1BitMessageBuffer

This class handles handles decoding where the decode buffer is viewed as a set of bits (i.e. a bit offset is maintained).

Field Summary

Asn1PerTraceHandler	mTraceHandler

Fields inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer
mByteCount

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1DecodeBitBuffer(byte[] msgdata) This constructor creates a decode buffer object that references an encoded ASN.1 message.
public	Asn1DecodeBitBuffer(java.io.InputStream istream) This constructor creates a decode buffer object that references an encoded ASN.1 message.

Method Summary

void	binDump(java.io.PrintStream out,java.lang.String varName) This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.

void	`binDump(java.lang.String varName)` This method invokes an overloaded version of binDump to dump the encoded message to standard output.
void	`byteAlign()` This methods byte-aligns the buffer.
boolean	`decodeBit()` This method decodes a single bit value.
int	`decodeBitsToInt(int nbits)` This method decodes bits from the input stream into a standard integer value.
long	`decodeBitsToLong(int nbits)` This method decodes bits from the input stream into a long integer value.
void	`decodeBitsToOctetArray(byte[] value,int offset,int nbits)` This method decodes bits from the input stream into an array of octets.
void	`decodeBitsToOctetArray(byte[] value,int offset,int bitOffset,int nbits)` This method decodes bits from the input stream into an array of octets.
int	`getAvailableBits()` Return a number of bits that can definitely be read from this buffer without causing an end of buffer exception.
long	`getBitOffset()` This method returns the absolute offset to the current bit in the decode buffer.
int	`getBitOffsetInByte()` Return the bit offset, within the byte, of the next bit to be read.
int	`getMsgBitCnt()` This method returns the number of bits in the encoded PER message.
Asn1PerTraceHandler	`getTraceHandler()` This method will return a reference to the internal trace handler object used to trace the bit fields within a PER message.
boolean	`hasMoreBits()` This method returns true if there are more bits to be read from this buffer.
void	`mark(int readLimit)` This method is used to mark the current position in the input stream for retry processing.
void	`moveBitCursor(long offset)` This method moves the bit cursor to the given offset.
int	`readByte()` This method returns the next available 8-bit value from the input stream.
void	`reset()` This method is used to reset the current position in the input stream back to the location of the last 'mark' call.

void	setInputStream(byte[] msgdata,int offset,int length) This method will set the input stream from which data is read.
void	skipBits(long bits) Skip the given number of bits.

Methods inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer
addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mTraceHandler

protected com.objsys.asn1j.runtime.Asn1PerTraceHandler **mTraceHandler**

Constructors

Asn1DecodeBitBuffer

public **Asn1DecodeBitBuffer**(byte[] msgdata)

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

Parameters:

msgdata - Byte array containing an encoded ASN.1 message.

Asn1DecodeBitBuffer

public **Asn1DecodeBitBuffer**(java.io.InputStream istream)

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

Parameters:

istream - Input stream containing an encoded ASN.1 message.

Methods

getTraceHandler

```
public Asn1PerTraceHandler getTraceHandler()
```

This method will return a reference to the internal trace handler object used to trace the bit fields within a PER message.

---

binDump

```
public void binDump(java.lang.String varName)
```

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

Parameters:

`varName` - Name of top-level message object variable

---

binDump

```
public void binDump(java.io.PrintStream out,
         java.lang.String varName)
```

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

Parameters:

`out` - PrintStream object to which output should be written

`varName` - Name of top-level message object variable

---

byteAlign

```
public void byteAlign()
```

This methods byte-aligns the buffer.

---

decodeBit

```
public boolean decodeBit()
  throws Asn1EndOfBufferException,
         java.io.IOException
```

This method decodes a single bit value.

Returns:

Boolean value of bit that was decoded.

---

decodeBitsToLong

```
public long decodeBitsToLong(int nbits)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nbits` - Number of bits to decode

Returns:

Long integer value containing decoded bits

decodeBitsToInt

```
public int decodeBitsToInt(int nbits)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nbits` - Number of bits to decode

Returns:

Integer value containing decoded bits

decodeBitsToOctetArray

```
public void decodeBitsToOctetArray(byte[] value,
          int offset,
          int nbits)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`value` - Octet array for decoded data
`offset` - Starting byte offset into array
`nbits` - Number of bits to decode

decodeBitsToOctetArray

```
public void decodeBitsToOctetArray(byte[] value,
          int offset,
          int bitOffset,
          int nbits)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes bits from the input stream into an array of octets. The user is expected to have provided an array large enough to hold the number of bits requested to be decoded. The first bit is decoded into the given offset byte at the MSB if bitOffset == 0, and at the LSB if bitOffset == 7.

Parameters:

`value` - Octet array for decoded data
`offset` - Starting byte offset into array
`bitOffset` - Where in first byte the first bit goes
`nbits` - Number of bits to decode

getBitOffsetInByte

```
public final int getBitOffsetInByte()
```

Return the bit offset, within the byte, of the next bit to be read. The returns 7 for the most significant bit, and 0 for the least significant bit.

Returns:

(continued from last page)

getBitOffset

```
public final long getBitOffset()
```

This method returns the absolute offset to the current bit in the decode buffer. This returns 0 when the first bit of the message is the next bit to be decoded.

Returns:

Offset to current bit in decode buffer

---

getMsgBitCnt

```
public final int getMsgBitCnt()
```

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

Returns:

Count of bits in encoded message

---

getAvailableBits

```
public final int getAvailableBits()
```

Return a number of bits that can definitely be read from this buffer without causing an end of buffer exception. This only returns zero when the end of the buffer has been reached (or there is some error trying to read from the stream). There may be many more bits available in the underlying stream than what is returned by this function. This might need to read a byte from the underlying stream to determine whether there are more bytes or not. If it cannot read from the stream, this will return 0.

Returns:

---

hasMoreBits

```
public final boolean hasMoreBits()
```

This method returns true if there are more bits to be read from this buffer. This might need to read a byte from the underlying stream to determine whether there are more bytes or not. If it cannot do so, this will return false;

---

mark

```
public void mark(int readLimit)
```

This method is used to mark the current position in the input stream for retry processing. It is equivalent to the Java InputStream 'mark' method.

Parameters:

`readLimit` - Max number of bytes that can be read before mark becomes invalid.

---

moveBitCursor

```
public final void moveBitCursor(long offset)
  throws Asn1EndOfBufferException,
         java.io.IOException
```

This method moves the bit cursor to the given offset.

Parameters:

`offset` - Absolute bit offset value

(continued from last page)

readByte

```
public final int readByte()
  throws Asn1Exception,
         java.io.IOException
```

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

Returns:

Next 8-bit byte value from input stream

---

reset

```
public 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 the Java InputStream 'reset' method. This does NOT restore the container stack to its state at the last mark(). It seems unlikely the container stack would be changed between a mark() and a reset().

---

setInputStream

```
public final void setInputStream(byte[] msgdata,
          int offset,
          int length)
```

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

Parameters:

`msgdata` - Byte array containing encoded message data
`offset` - Starting offset of data in the byte array
`length` - Length (in bytes) of the encoded data

---

skipBits

```
public final void skipBits(long bits)
  throws java.io.IOException,
         Asn1EndOfBufferException
```

Skip the given number of bits. This many bits must be present or else this will throw Asn1EndOfBufferException.

Parameters:

`bits`

Throws:

`IOException`
`Asn1EndOfBufferException`

com.objsys.asn1j.runtime
Class Asn1DecodeBuffer

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |-com.objsys.asn1j.runtime.Asn1MessageBuffer
         |-com.objsys.asn1j.runtime.Asn1DecodeBuffer
```

Direct Known Subclasses:
Asn1MderDecodeBuffer, Asn1DecodeBitBuffer, Asn1BerDecodeBuffer

---

public abstract class **Asn1DecodeBuffer**
extends Asn1MessageBuffer

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

Field Summary

int	mByteCount

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Method Summary

void	addCaptureBuffer(java.io.ByteArrayOutputStream buffer) This method is used to add a capture buffer to the internal capture buffer list.
void	capture(int nbytes) This method captures bytes from the input stream to a separate object for later analysis.
long	decodeIntValue(int length,boolean signExtend) This method decodes the contents of an ASN.1 integer value.
int[]	decodeOIDContents(int llen) This method decodes the contents of an ASN.1 object identifier value.
int[]	decodeRelOIDContents(int llen) This method decodes the contents of an ASN.1 relative object identifier value.
int	getByteCount() This method returns the count of bytes currently read from the message being decoded.
java.io.InputStream	getInputStream() This method returns a reference to the current current decode input stream object.

boolean	`getLazyOpenTypeDecode()` Return true if lazy open type decoding is on.
void	`hexDump()` This method provides a hex dump of the bytes in the message being decoded.
void	`init()` This method initializes the input stream for decoding.
void	`mark(int readLimit)` This method is used to mark the current position in the input stream for retry processing.
int	`read()` The read method reads a single byte from the current input stream and returns it to the caller.
void	`read(byte[] buffer)` This version of the read method reads the number of bytes equal to the length of the given input buffer.
void	`read(byte[] buffer,int offset,int nbytes)` 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.
int	`read2Bytes()` Read the next two bytes from the current input stream into an int, and return that int.
int	`read4Bytes()` Read the next four bytes from the current input stream into an int, and return that int.
abstract int	`readByte()` This abstract method returns the next available 8-bit value from the input stream.
void	`removeCaptureBuffer(java.io.ByteArrayOutputStream buffer)` This method is used to remove a capture buffer from the internal capture buffer list.
void	`reset()` This method is used to reset the current position in the input stream back to the location of the last 'mark' call.
void	`setInputStream(byte[] msgdata,int offset,int length)` This method will set the input stream from which data is read.
void	`setLazyOpenTypeDecode(boolean value)` This method turns lazy open type decoding on or off.
long	`skip(long nbytes)` This method will skip over the requested number of bytes in the input stream.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBuffer`
`addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`

getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mByteCount

```
protected int mByteCount
```

Methods

capture

```
public void capture(int nbytes)
  throws Asn1EndOfBufferException,
         java.io.IOException
```

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

Parameters:

`nbytes` - Number of bytes to capture

decodeIntValue

```
public long decodeIntValue(int length,
          boolean signExtend)
  throws Asn1Exception,
         java.io.IOException
```

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

decodeOIDContents

```
public int[] decodeOIDContents(int llen)
  throws Asn1Exception,
         java.io.IOException
```

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

decodeRelOIDContents

```
public int[] decodeRelOIDContents(int llen)
  throws Asn1Exception,
         java.io.IOException
```

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

addCaptureBuffer

```
public void addCaptureBuffer(java.io.ByteArrayOutputStream buffer)
```

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

Parameters:

`buffer` - Buffer into which captured bytes are to be stored

removeCaptureBuffer

```
public void removeCaptureBuffer(java.io.ByteArrayOutputStream buffer)
```

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

Parameters:

`buffer` - Buffer in which captured bytes stored

getByteCount

```
public int getByteCount()
```

This method returns the count of bytes currently read from the message being decoded.

Returns:

Count of bytes read from input stream

getInputStream

```
public java.io.InputStream getInputStream()
```

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

Returns:

New input stream object containing encoded message

getLazyOpenTypeDecode

```
public boolean getLazyOpenTypeDecode()
```

Return true if lazy open type decoding is on.

Returns:

true if lazy open type decoding is on

hexDump

```
public void hexDump()
```

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

init

```
protected void init()
```

This method initializes the input stream for decoding.

mark

```
public void mark(int readLimit)
```

This method is used to mark the current position in the input stream for retry processing. It is equivalent to the Java InputStream 'mark' method.

Parameters:

`readLimit` - Max number of bytes that can be read before mark becomes invalid.

read2Bytes

```
public final int read2Bytes()
  throws Asn1EndOfBufferException,
         java.io.IOException
```

Read the next two bytes from the current input stream into an int, and return that int. The bytes of the int, from lowest to highest, will correspond to the bytes read from the stream, from last to first. The highest two bytes will be 0. Each byte read will be written to all registered capture buffers.

Returns:

an int representing the 2 bytes read, as described above.

Throws:

`Asn1EndOfBufferException` - if at end-of-stream

`java.io.IOException` - Java I/O error

read4Bytes

```
public final int read4Bytes()
  throws Asn1EndOfBufferException,
         java.io.IOException
```

Read the next four bytes from the current input stream into an int, and return that int. The bytes of the int, from lowest to highest, will correspond to the bytes read from the stream, from last to first. Each byte read will be written to all registered capture buffers.

Returns:

an int representing the 4 bytes read, as described above.

Throws:

`Asn1EndOfBufferException` - if at end-of-stream

`java.io.IOException` - Java I/O error

read

```
public final int read()
  throws Asn1EndOfBufferException,
         java.io.IOException
```

(continued from last page)

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.

Returns:

byte that was read from the input stream

Throws:

`Asn1EndOfBufferException` - if at end-of-stream
`java.io.IOException` - Java I/O error

---

readByte

```
public abstract int readByte()
  throws Asn1Exception,
         java.io.IOException
```

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

---

read

```
public final void read(byte[] buffer,
          int offset,
          int nbytes)
  throws Asn1EndOfBufferException,
         java.io.IOException
```

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.

Parameters:

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

Throws:

`Asn1EndOfBufferException` - if at end-of-stream
`java.io.IOException` - Java I/O error

---

read

```
public void read(byte[] buffer)
  throws Asn1EndOfBufferException,
         java.io.IOException
```

This version of the read method reads the number of bytes equal to the length of the given input buffer.

Parameters:

`buffer` - the buffer into which the data is read

Throws:

`Asn1EndOfBufferException` - if at end-of-stream
`java.io.IOException` - Java I/O error

---

reset

```
public 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 the Java InputStream 'reset' method.

setLazyOpenTypeDecode

```
public void setLazyOpenTypeDecode(boolean value)
```

This method turns lazy open type decoding on or off. Its setting is relevant only when generating table constraint code (otherwise, open types cannot be decoded). Generated decode methods check this setting to determine whether to decode open types or not. When lazy open type decoding is turned on, you can use the generated decodeOpenType* methods to decode open types (again, assuming table constraint code was generated).

setInputStream

```
public void setInputStream(byte[] msgdata,
          int offset,
          int length)
```

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

Parameters:

`msgdata` - Byte array containing encoded message data
`offset` - Starting offset of data in the byte array
`length` - Length (in bytes) of the encoded data

skip

```
public long skip(long nbytes)
  throws java.io.IOException
```

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

Parameters:

`nbytes` - Number of bytes to skip

com.objsys.asn1j.runtime
Class Asn1DerDecodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
               ┗-com.objsys.asn1j.runtime.Asn1DerDecodeBuffer
```

Direct Known Subclasses:
Asn1DerInputStream

---

public class **Asn1DerDecodeBuffer**
extends Asn1BerDecodeBuffer

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1DerDecodeBuffer(byte[] msgdata)` This constructor creates a DER decode buffer object that references an encoded ASN.1 message.
public	`Asn1DerDecodeBuffer(java.io.InputStream istream)` This constructor creates a DER decode buffer object that references an encoded ASN.1 message.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1BerDecodeBuffer`
`calcIndefLen, decodeEnumValue, decodeEnumValue, decodeLength, decodeOpenType, decodeOpenType, decodeTag, decodeTagAndLength, getLastTag, matchTag, matchTag, matchTag, movePastEOC, parse, peekTag, peekTag, readByte`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip`

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1DerDecodeBuffer

`public` **`Asn1DerDecodeBuffer`**`(byte[] msgdata)`

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

Parameters:

`msgdata` - Byte array containing an encoded ASN.1 message.

Asn1DerDecodeBuffer

`public` **`Asn1DerDecodeBuffer`**`(java.io.InputStream istream)`

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

Parameters:

`istream` - Input stream containing an encoded ASN.1 message.

com.objsys.asn1j.runtime
Class Asn1DerEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev
               ┗-com.objsys.asn1j.runtime.Asn1BerEncodeBuffer
                  ┗-com.objsys.asn1j.runtime.Asn1DerEncodeBuffer
```

public class **Asn1DerEncodeBuffer**
extends Asn1BerEncodeBuffer

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

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev
mByteIndex, mData

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
INITIAL_SIZE

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1DerEncodeBuffer() This constructor creates a DER encode buffer object with the default size.
public	Asn1DerEncodeBuffer(int size) This constructor creates a DER encode buffer object with the given initial size.

Method Summary

boolean	isBER() This method returns true when the underlying buffer is a BER buffer.

Methods inherited from class com.objsys.asn1j.runtime.Asn1BerEncodeBuffer
binDump, binDump, checkSize, encodeIdentifier, encodeIntValue, encodeLength, encodeTag, encodeTag, encodeTagAndLength, encodeTagAndLength, isBER

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev
checkSize, copy, copy, copy, copy, getByteArrayInputStream, getMsgCopy, getMsgLength, initBuffer, reset, toString, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1DerEncodeBuffer

`public` **`Asn1DerEncodeBuffer`**`()`

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

Asn1DerEncodeBuffer

`public` **`Asn1DerEncodeBuffer`**`(int size)`

This constructor creates a DER encode buffer object with the given initial size. Whenever the buffer becomes full, the buffer will be expanded. For best performance, this size should be large enough to prevent resizing in normal operation.

Parameters:

`size` - The initial size in bytes of an encode buffer.

Methods

isBER

`public boolean` **`isBER`**`()`

This method returns true when the underlying buffer is a BER buffer. This returns false for the DER implementation.

Returns:

False.

com.objsys.asn1j.runtime
Class Asn1DerInputStream

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
               ┗-com.objsys.asn1j.runtime.Asn1DerDecodeBuffer
                  ┗-com.objsys.asn1j.runtime.Asn1DerInputStream
```

All Implemented Interfaces:
Asn1InputStream

---

public class **Asn1DerInputStream**
extends Asn1DerDecodeBuffer
implements Asn1InputStream

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1DerInputStream(java.io.InputStream istream)` This constructor creates a DER decode buffer object that references an encoded ASN.1 message.

Method Summary

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.
void	`close()` Closes this input stream and releases any system resources associated with the stream.
void	`mark(int readLimit)` This method is used to mark the current position in the input stream for retry processing.

boolean	markSupported() Tests if this input stream supports the markand resetmethods.
void	reset() This method is used to reset the current position in the input stream back to the location of the last 'mark' call.
long	skip(long nbytes) This method will skip over the requested number of bytes in the input stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1BerDecodeBuffer
calcIndefLen, decodeEnumValue, decodeEnumValue, decodeLength, decodeOpenType, decodeOpenType, decodeTag, decodeTagAndLength, getLastTag, matchTag, matchTag, matchTag, movePastEOC, parse, peekTag, peekTag, readByte

Methods inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer
addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1DerInputStream

public **Asn1DerInputStream**(java.io.InputStream istream)

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

Parameters:

istream - Input stream containing an encoded ASN.1 message.

Methods

(continued from last page)

available

```
public int available()
  throws java.io.IOException
```

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.

Throws:

`IOException` - if an I/O error occurs.

---

close

```
public void close()
  throws java.io.IOException
```

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

Throws:

`IOException` - if an I/O error occurs.

---

mark

```
public void mark(int readLimit)
```

This method is used to mark the current position in the input stream for retry processing. It is equivalent to the Java InputStream 'mark' method.

Parameters:

`readLimit` - Max number of bytes that can be read before mark becomes invalid.

---

markSupported

```
public boolean markSupported()
```

Tests if this input stream supports the `mark`and `reset`methods. The `markSupported`method of `InputStream`returns `false`.

Returns:

`true`if this true type supports the mark and reset method; `false`otherwise.

---

reset

```
public 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 the Java InputStream 'reset' method.

---

skip

```
public long skip(long nbytes)
  throws java.io.IOException
```

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

Parameters:

`nbytes` - Number of bytes to skip

com.objsys.asn1j.runtime
Class Asn1DiscreteCharSet

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1CharSet
      |-com.objsys.asn1j.runtime.Asn1DiscreteCharSet
```

public class **Asn1DiscreteCharSet**
extends Asn1CharSet

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1CharSet`
`mABitsPerChar`, `mUBitsPerChar`

Constructor Summary

public	`Asn1DiscreteCharSet(java.lang.String charSet)` This constructor sets the permitted alphabet chracter set
public	`Asn1DiscreteCharSet(int[] charSet)` This constructor sets the permitted alphabet chracter set

Method Summary

int	`getCharAtIndex(int index)` This method will fetch the character from the permitted alphabet at the given index.
int	`getCharIndex(int charValue)` This method will determine the index of the given character within the permitted alphabet character set.
int	`getMaxValue()` This method will determine the maximum value of the given character within the permitted alphabet character set.
boolean	`helpValidate(char c)` This function helps validate a character string by checking a character to see if it is in the character set.
boolean	`validate(java.lang.String s)` This method will validate a character string by comparing its contents to the character set.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharSet`
`getCharAtIndex, getCharIndex, getMaxValue, getNumBitsPerChar, validate`

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1DiscreteCharSet

`public` **`Asn1DiscreteCharSet`**`(java.lang.String charSet)`

This constructor sets the permitted alphabet chracter set

Parameters:

`charSet` - Permitted alphabet character set

Asn1DiscreteCharSet

`public` **`Asn1DiscreteCharSet`**`(int[] charSet)`

This constructor sets the permitted alphabet chracter set

Parameters:

`charSet` - Permitted alphabet character set

Methods

getCharAtIndex

`public int` **`getCharAtIndex`**`(int index)`
`throws Asn1ConsVioException`

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

Throws:

`Asn1ConsVioException` - Index not within define range

getCharIndex

`public int` **`getCharIndex`**`(int charValue)`
`throws Asn1ConsVioException`

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

Throws:

`Asn1ConsVioException` - Character not found in set

getMaxValue

`public int` **`getMaxValue`**`()`

This method 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.

Returns:

Upper Bound Character or Character with max int value

validate

`public boolean` **`validate`**`(java.lang.String s)`

This method will validate a character string by comparing its contents to the character set. If a character string contains characters that are not in the character set, this method will return false. Otherwise it returns true.

Parameters:

`s` - The string to be validated.

Returns:

False if the string contains invalid characters; true otherwise.

helpValidate

`protected boolean` **`helpValidate`**`(char c)`

This function helps validate a character string by checking a character to see if it is in the character set. It returns false if a character is not contained in the set and true otherwise.

com.objsys.asn1j.runtime
Class Asn1Duration

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1Duration
```

public class **Asn1Duration**
extends Asn18BitCharString

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 34).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString

BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Duration() The default constructor creates an empty string object.
public	Asn1Duration(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 DURATION string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 DURATION value in accordance with the octet encoding rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 DURATION value in accordance with the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 DURATION string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes and writes to the stream an ASN.1 DURATION value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 DURATION value in accordance with the octet encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 DURATION value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 DURATION value in accordance with the packed encoding rules (PER) directly into the stream.

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`

`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 34).

Constructors

Asn1Duration

```
public Asn1Duration()
```

The default constructor creates an empty string object.

Asn1Duration

```
public Asn1Duration(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
                   java.io.IOException
```

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

Parameters:

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

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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

Parameters:

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

Returns:

Length in octets of encoded component

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`java.io.IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 DURATION value in accordance with the octet encoding rules (OER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 DURATION value in accordance with the octet encoding rules (PER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 DURATION value in accordance with the packed encoding rules (PER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 DURATION value in accordance with the packed encoding rules (PER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 DURATION 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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object

Throws:

`java.io.IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1ElementTracker

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1ElementTracker
```

All Implemented Interfaces:
Asn1NamedEventHandler

---

public class **Asn1ElementTracker**
extends java.lang.Object
implements Asn1NamedEventHandler

Maintains an element stack so that a full name for the current element can be obtained. This simply pushes/pops elements on/off the element stack when the event handler methods are invoked.

Constructor Summary

public	Asn1ElementTracker()

Method Summary

void	characters(java.lang.String svalue,short typeCode)
void	endElement(java.lang.String name,int index)
java.lang.String	getCurrentElement()
void	startElement(java.lang.String name,int index)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

Asn1ElementTracker

public **Asn1ElementTracker**()

Methods

(continued from last page)

characters

```
public void characters(java.lang.String svalue,
          short typeCode)
```

---

endElement

```
public void endElement(java.lang.String name,
          int index)
```

---

startElement

```
public void startElement(java.lang.String name,
          int index)
```

---

getCurrentElement

```
public java.lang.String getCurrentElement()
```

com.objsys.asn1j.runtime
Class Asn1EncodeBitBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeBitBuffer
```

All Implemented Interfaces:
Asn1BitMessageBuffer

Direct Known Subclasses:
Asn1PerEncodeBuffer, Asn1OerEncodeBuffer, Asn1NasEncodeBuffer

---

public class **Asn1EncodeBitBuffer**
extends Asn1EncodeBuffer
implements Asn1BitMessageBuffer

An encoding buffer that is bit-oriented. This is useful when dealing with encoding rules where individual bits need to be written, such as PER.

Field Summary

int	mByteIndex
byte[]	mData
Asn1PerTraceHandler	mTraceHandler

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
INITIAL_SIZE

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1EncodeBitBuffer() The default constructor initializes the encode buffer to the default buffer size (INITIAL_SIZE bytes).
public	Asn1EncodeBitBuffer(int size) This constructor allows specification of the initial size.

Method Summary

void	`binDump(java.io.PrintStream out,java.lang.String varName)` This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.
void	`byteAlign()` Byte-align the buffer, possibly conditionally.
void	`checkSize(int bytesRequired)` This method determines if the encode buffer can hold the requested number of additional bytes.
void	`copy(byte value)` This method is used to copy a single byte to the encode buffer.
void	`copy(byte[] value,int offset,int len)` This method copies multiple bytes from the given array to the encode buffer.
void	`encodeBit(boolean value)` This method encodes a single bit value.
void	`encodeBits(byte[] value,int offset,int nbits)` This method encodes bit values from an array of octets.
void	`encodeBits(byte[] value,int offset,int bitOffset,int nbits)` This method encodes bit values from an array of octets.
void	`encodeBits(byte value,int nbits)` This method encodes bit values from an octet.
void	`encodeLongBits(long value,int nbits)` This method encodes bit values from a long.
void	`encodeLongBits(long value,int nbits,boolean merge)` This method encodes bit values from a long.
int	`getBitOffsetInByte()` Return the bit offset of the next bit to be encoded.
byte[]	`getBuffer()` This method returns a reference to the byte buffer used to hold the encoded message.
java.io.ByteArrayInputStream	`getByteArrayInputStream()` This method returns a reference to a byte array input stream representing the encoded message.
int	`getByteIndex()` This method returns the current byte index into the encode buffer.
int	`getMsgBitCnt()` This method returns the number of bits in the encoded message.

int	getMsgByteCnt() This method returns the number of bytes written to the buffer.
byte[]	getMsgCopy() This method returns the encoded message in a byte array.
int	getMsgLength() This method returns the length (in bytes) of the encoded message component.
Asn1PerTraceHandler	getTraceHandler() This method will return a reference to the internal trace handler object used to trace the bit fields within a PER message.
void	hexDump() This method dumps the encoded message in hex/ascii format to the standard output stream.
boolean	isByteAligned() Return true/false indicating whether a multiple of 8 bits has been encoded thus far.
void	reset() This method resets the buffer object so that it can be reused to encode another message.
void	reverseBytes(int offset,int nbytes) This method reverses a series of bytes at a given offset within the encode buffer.
void	setMsgBitCnt(int count) This method sets the position in the encoded message such that getMsgBitCnt will subsequently return the given number.
java.lang.String	toString() This method will return a string representation of the data in the encode buffer.
void	write(java.io.OutputStream out) This method writes the encoded record to the given output stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mByteIndex

`protected int` **`mByteIndex`**

mData

`protected byte` **`mData`**

mTraceHandler

`protected com.objsys.asn1j.runtime.Asn1PerTraceHandler` **`mTraceHandler`**

Constructors

Asn1EncodeBitBuffer

`public` **`Asn1EncodeBitBuffer`**`()`

The default constructor initializes the encode buffer to the default buffer size (INITIAL_SIZE bytes).

Asn1EncodeBitBuffer

`public` **`Asn1EncodeBitBuffer`**`(int size)`

This constructor allows specification of the initial size. This overrides the INITIAL_SIZE constant value.

Parameters:

`size -` initial Buffer size

Methods

byteAlign

`public void` **`byteAlign`**`()`

Byte-align the buffer, possibly conditionally. This class implements this method to unconditionally byte-align. Subclasses, however, may override this (as PER does) to only byte-align under certain conditions (such as when aligned PER is specified).

isByteAligned

`public final boolean` **`isByteAligned`**`()`

Return true/false indicating whether a multiple of 8 bits has been encoded thus far.

getTraceHandler

`public Asn1PerTraceHandler` **`getTraceHandler`**`()`

(continued from last page)

This method will return a reference to the internal trace handler object used to trace the bit fields within a PER message.

binDump

```
public void binDump(java.io.PrintStream out,
         java.lang.String varName)
```

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

checkSize

```
protected final void checkSize(int bytesRequired)
```

This method determines if the encode buffer can hold the requested number of additional bytes. If not, the buffer is expanded. This implementation assumes mByteIndex is used, and that the array is filling from 0..n.

Parameters:

`bytesRequired` - Number of required bytes.

copy

```
public void copy(byte value)
```

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

Parameters:

`value` - The byte value to copy

copy

```
public void copy(byte[] value,
         int offset,
         int len)
  throws Asn1Exception
```

This method copies multiple bytes from the given array to the encode buffer.

Parameters:

`value`

`offset` - The first byte copied is value[offset]

`len` - The last byte copied is value[offset + len - 1]

Throws:

`Asn1Exception`

getBitOffsetInByte

```
public final int getBitOffsetInByte()
```

Return the bit offset of the next bit to be encoded. This returns 7 for the most significant bit, and 0 for the least significant bit.

Returns:

getBuffer

```
public byte[] getBuffer()
```

This method returns a reference to the byte buffer used to hold the encoded message. (This is dangerous!)

Returns:

(continued from last page)

Byte buffer reference

getByteArrayInputStream

```
public java.io.ByteArrayInputStream getByteArrayInputStream()
```

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

Returns:

byte array input stream containing encoded message

getMsgBitCnt

```
public final int getMsgBitCnt()
```

This method returns the number of bits in the encoded message.

Returns:

Count of bits in encoded message

setMsgBitCnt

```
public final void setMsgBitCnt(int count)
```

This method sets the position in the encoded message such that getMsgBitCnt will subsequently return the given number.

Parameters:

`count` - The number of bits. This many bits must fit within the current buffer.

getMsgCopy

```
public final byte[] getMsgCopy()
```

This method returns 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.

Returns:

byte array containing encoded message

encodeBit

```
public void encodeBit(boolean value)
```

This method encodes a single bit value.

Parameters:

`value` - Boolean value of bit to be encoded.

encodeLongBits

```
public final void encodeLongBits(long value,
                                 int nbits)
```

This method encodes bit values from a long. The least significant nbits bits of the long value are encoded.

encodeLongBits

```
public final void encodeLongBits(long value,
                                 int nbits,
                                 boolean merge)
```

This method encodes bit values from a long. The least significant nbits bits of the long value are encoded.

(continued from last page)

Parameters:

`value` - The value to take bits from

`nbits` - The number of bits to encode

`merge` - If true, preserve bits that follow the encoded bits in any bytes that are touched. This can be set when overwriting certain bits within an encoding.

---

encodeBits

```
public final void encodeBits(byte value,
         int nbits)
  throws Asn1InvalidArgException
```

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

Parameters:

`value` - Octet containing bits to be encoded

`nbits` - Number of bits to encode

---

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int bitOffset,
         int nbits)
  throws Asn1InvalidArgException
```

This method encodes bit values from an array of octets.

Parameters:

`value` - Octet array containing bits to be encoded

`offset` - Starting byte offset in value

`bitOffset` - Starting bit offset in first byte. Pass 0 to begin encoding with the most significant bit. Pass 7 to begin with the least significant bit.

`nbits` - Number of bits to encode

---

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int nbits)
  throws Asn1InvalidArgException
```

This method encodes bit values from an array of octets.

Parameters:

`value` - Octet array containing bits to be encoded

`offset` - Starting byte offset in value

`nbits` - Number of bits to encode

---

getByteIndex

```
public int getByteIndex()
```

This method returns the current byte index into the encode buffer.

Returns:

Byte index value

---

getMsgByteCnt

```
public int getMsgByteCnt()
```

(continued from last page)

This method returns the number of bytes written to the buffer. If the last byte written to is a partial byte, it is counted as a full byte.

Returns:

Count of bytes in encoded message

---

getMsgLength

```
public int getMsgLength()
```

This method returns the length (in bytes) of the encoded message component.

Returns:

length of encoded message component

---

reset

```
public void reset()
```

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

---

reverseBytes

```
public void reverseBytes(int offset,
          int nbytes)
```

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

Parameters:

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

---

write

```
public final void write(java.io.OutputStream out)
  throws java.io.IOException
```

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

Parameters:

`out` - Output stream to which record is to be written

---

toString

```
public java.lang.String toString()
```

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

Returns:

Stringified representation of the value

---

hexDump

```
public void hexDump()
```

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

com.objsys.asn1j.runtime
Class Asn1EncodeBuffer

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |-com.objsys.asn1j.runtime.Asn1MessageBuffer
         |-com.objsys.asn1j.runtime.Asn1EncodeBuffer
```

Direct Known Subclasses:
Asn1EncodeByteBufferRev, Asn1EncodeByteBuffer, Asn1EncodeBitBuffer

---

public abstract class **Asn1EncodeBuffer**
extends Asn1MessageBuffer

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

Field Summary

static final int	INITIAL_SIZE This constant specifies the default initial size of the encode buffer. Value: **1024**

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1EncodeBuffer()

Method Summary

abstract void	binDump(java.io.PrintStream out,java.lang.String varName) This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.
void	binDump(java.lang.String varName) This method invokes an overloaded version of binDump to dump the encoded message to standard output.
abstract void	copy(byte value) This abstract method is used to copy a single byte to the encode buffer.
void	copy(byte[] value) This method copies multiple bytes to the encode buffer

abstract void	copy(byte[] value,int offset,int len) This method copies multiple bytes from the given array to the encode buffer.
static int	encodeIntSigned(long value,byte[] dest,int offset) Encode an integer value as a signed value (2's complement), in the minimum number of octets possible (>=1), into the given array.
static int	encodeIntUnsigned(long value,byte[] dest,int offset) Encode an integer value as an unsigned value (binary integer), in the minimum number of octets possible (>=1), into the given array.
abstract java.io.ByteArrayInputStream	getByteArrayInputStream() This method returns a reference to a byte array input stream representing the encoded message.
java.io.InputStream	getInputStream() This method returns an input stream representing the encoded message.
static int	getMinimalOctetsSigned(long value) Return the minimal number of octets required to represent the given value, when a signed representation is being used.
static int	getMinimalOctetsUnsigned(long value) Return the minimal number of octets required to represent the given value, when an unsigned representation is being used.
abstract byte[]	getMsgCopy() This method returns the encoded message in a byte array.
abstract int	getMsgLength() This method returns the length (in bytes) of the encoded message component.
java.io.OutputStream	getOutputStream() Return an output stream associated with this object.
void	hexDump() This method dumps the encoded message in hex/ascii format to the standard output stream.
void	hexDump(java.io.PrintStream out) This method dumps the encoded message in hex/ascii format to the given print output stream.
abstract void	reset() This method resets the buffer to allow a new record to be encoded into it.
abstract void	write(java.io.OutputStream out) This method writes the encoded record to the given output stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase

getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

INITIAL_SIZE

public static final int **INITIAL_SIZE**

This constant specifies the default initial size of the encode buffer. It is currently set to 1024 bytes.

Constructors

Asn1EncodeBuffer

public **Asn1EncodeBuffer**()

Methods

binDump

public void **binDump**(java.lang.String varName)

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

Parameters:

varName - Variable name to include in output.

binDump

public abstract void **binDump**(java.io.PrintStream out,
java.lang.String varName)

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

Parameters:

out - stream to print to.

varName - Variable name to include in output.

copy

public abstract void **copy**(byte value)

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

Parameters:

value - The byte value to copy

(continued from last page)

copy

```
public void copy(byte[] value)
  throws Asn1Exception
```

This method copies multiple bytes to the encode buffer

Parameters:

`value` - Array of bytes to copy to the encode buffer

---

copy

```
public abstract void copy(byte[] value,
          int offset,
          int len)
  throws Asn1Exception
```

This method copies multiple bytes from the given array to the encode buffer.

Parameters:

`value` - The bytes to be encoded.
`offset` - The first byte copied is value[offset]
`len` - The last byte copied is value[offset + len - 1]

---

encodeIntSigned

```
public static int encodeIntSigned(long value,
          byte[] dest,
          int offset)
```

Encode an integer value as a signed value (2's complement), in the minimum number of octets possible (>=1), into the given array.

Parameters:

`value` - The value to encode.
`dest` - The array to but the encoding into. It must be large enough to hold the result, taking into account the given offset.
`offset` - The index where the first byte should go.

Returns:

The number of bytes.

---

encodeIntUnsigned

```
public static int encodeIntUnsigned(long value,
          byte[] dest,
          int offset)
```

Encode an integer value as an unsigned value (binary integer), in the minimum number of octets possible (>=1), into the given array.

Parameters:

`value` - The value to encode. It must be non-negative.
`dest` - The array to but the encoding into. It must be large enough to hold the result, taking into account the given offset.
`offset` - The index where the first byte should go.

Returns:

The number of bytes.

---

getByteArrayInputStream

```
public abstract java.io.ByteArrayInputStream getByteArrayInputStream()
```

(continued from last page)

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

Returns:

byte array input stream containing encoded message

---

getInputStream

`public final java.io.InputStream` **`getInputStream`**`()`

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

Returns:

Input stream containing encoded message

---

getOutputStream

`public final java.io.OutputStream` **`getOutputStream`**`()`

Return an output stream associated with this object. Anything written to the OutputStream will be written to this buffer using copy(byte) or copy(byte[]).

Returns:

The output sttream.

---

getMinimalOctetsSigned

`public static int` **`getMinimalOctetsSigned`**`(long value)`

Return the minimal number of octets required to represent the given value, when a signed representation is being used.

Parameters:

`value -` The value to determine the minimal octets for.

Returns:

The number of octets, <= 8.

---

getMinimalOctetsUnsigned

`public static int` **`getMinimalOctetsUnsigned`**`(long value)`

Return the minimal number of octets required to represent the given value, when an unsigned representation is being used.

Parameters:

`value -` Must be >=0

Returns:

The number of octets, <= 8.

---

getMsgCopy

`public abstract byte[]` **`getMsgCopy`**`()`

This method returns 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.

Returns:

byte array containing encoded message

(continued from last page)

getMsgLength

```
public abstract int getMsgLength()
```

This method returns the length (in bytes) of the encoded message component.

Returns:

length of encoded message component

---

hexDump

```
public void hexDump()
```

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

---

hexDump

```
public void hexDump(java.io.PrintStream out)
```

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

Parameters:

`out` - Output stream object reference

---

reset

```
public abstract void reset()
```

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

---

write

```
public abstract void write(java.io.OutputStream out)
  throws java.io.IOException
```

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

Parameters:

`out` - Output stream to which record is to be written

Throws:

`java.io.IOException` - for any I/O error.

com.objsys.asn1j.runtime
Class Asn1EncodeByteBuffer

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |-com.objsys.asn1j.runtime.Asn1MessageBuffer
         |-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            |-com.objsys.asn1j.runtime.Asn1EncodeByteBuffer
```

Direct Known Subclasses:
Asn1XmlEncodeBuffer, Asn1XerEncodeBuffer

---

public abstract class **Asn1EncodeByteBuffer**
extends Asn1EncodeBuffer

An encoding buffer that is byte-oriented. This is usefule for encodings such as XML, where no part of the encoding consists of partial bytes. This is much the same as Asn1EncodeBitBuffer except that no bit offset needs to be taken into account.

Field Summary

int	mByteIndex
byte[]	mData

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
INITIAL_SIZE

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1EncodeByteBuffer() The default constructor initializes the encode buffer to the default buffer size (INITIAL_SIZE bytes).
public	Asn1EncodeByteBuffer(int size) This constructor allows specification of the initial size.

Method Summary

void	checkSize(int bytesRequired) This method determines if the encode buffer can hold the requested number of bytes.

void	copy(byte value) This method is used to copy a single byte to the encode buffer.
void	copy(byte[] value) This method copies multiple bytes to the encode buffer.
void	copy(byte[] value,int off,int len) This method copies multiple bytes to the encode buffer.
void	copyUnsafe(byte value) This method is used to copy a single byte to the encode buffer, without checking whether there is room or not.
byte[]	getBuffer() This method returns a reference to the byte buffer used to hold the encoded message.
java.io.ByteArrayInputStream	getByteArrayInputStream() This method returns a reference to a byte array input stream representing the encoded message.
byte[]	getMsgCopy() This method returns the encoded message in a byte array.
int	getMsgLength()
void	initBuffer(int size)
void	reset()
void	write(java.io.OutputStream out) This method writes the encoded record to the given output stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer

binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer

addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase

getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mByteIndex

```
protected int mByteIndex
```

mData

```
protected byte mData
```

Constructors

Asn1EncodeByteBuffer

```
public Asn1EncodeByteBuffer()
```

The default constructor initializes the encode buffer to the default buffer size (INITIAL_SIZE bytes).

Asn1EncodeByteBuffer

```
public Asn1EncodeByteBuffer(int size)
```

This constructor allows specification of the initial size. This overrides the INITIAL_SIZE constant value.

Parameters:

`size` - initial Buffer size

Methods

checkSize

```
protected void checkSize(int bytesRequired)
```

This method determines if the encode buffer can hold the requested number of bytes. If not, the buffer is expanded. This implementation is shared by the Xml and Xer subclasses. It assumes that the location at mByteIndex is not used, and that the array is filling from 0..n. Ber and Per override this method with slightly different behavior.

Parameters:

`bytesRequired` - Number of required bytes.

copy

```
public void copy(byte value)
```

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

Parameters:

`value` - The byte value to copy

copy

```
public void copy(byte[] value)
  throws Asn1Exception
```

(continued from last page)

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

Parameters:

`value` - Array of bytes to copy to the encode buffer

---

copy

```
public void copy(byte[] value,
          int off,
          int len)
  throws Asn1Exception
```

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

Parameters:

`value` - Array of bytes to copy to the encode buffer
`off` - Starting offset in array
`len` - The length to be encoded

---

copyUnsafe

```
protected void copyUnsafe(byte value)
```

This method is used to copy a single byte to the encode buffer, without checking whether there is room or not. The caller is responsible for ensuring the buffer is large enough. This can be useful to avoid multiple size checks when multiple bytes will be copied.

Parameters:

`value` - The byte value to copy

---

getByteArrayInputStream

```
public java.io.ByteArrayInputStream getByteArrayInputStream()
```

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

Returns:

byte array input stream containing encoded message

---

getBuffer

```
public byte[] getBuffer()
```

This method returns a reference to the byte buffer used to hold the encoded message.

Returns:

Byte buffer reference

---

getMsgCopy

```
public final byte[] getMsgCopy()
```

This method returns 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.

Returns:

byte array containing encoded message

(continued from last page)

getMsgLength

```
public int getMsgLength()
```

initBuffer

```
protected void initBuffer(int size)
```

reset

```
public void reset()
```

write

```
public final void write(java.io.OutputStream out)
  throws java.io.IOException
```

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

Parameters:

`out` - Output stream to which record is to be written

com.objsys.asn1j.runtime
Class Asn1EncodeByteBufferRev

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |-com.objsys.asn1j.runtime.Asn1MessageBuffer
         |-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            |-com.objsys.asn1j.runtime.Asn1EncodeByteBufferRev
```

Direct Known Subclasses:
Asn1BerEncodeBuffer

---

public abstract class **Asn1EncodeByteBufferRev**
extends Asn1EncodeBuffer

An encoding buffer that fills the underlying array in reverse order. This is useful for encoding rules like BER, where we encode fields in reverse order.

Field Summary

int	mByteIndex
byte[]	mData

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer

INITIAL_SIZE

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase

context, mTypeCode

Constructor Summary

public	Asn1EncodeByteBufferRev() The default constructor initializes the encode buffer to the default buffer size (INITIAL_SIZE bytes).
public	Asn1EncodeByteBufferRev(int size) This constructor allows specification of the initial size.

Method Summary

void	checkSize(int bytesRequired) This method determines if the encode buffer can hold the requested number of bytes.

void	copy(byte value) This method copies a single byte to the encode buffer.
void	copy(byte[] value) This method copies multiple bytes to the encode buffer
void	copy(byte[] value,int startOffset,int length) This method copies multiple bytes to the encode buffer
void	copy(java.lang.String value) This method copies a character string into the encode buffer
java.io.ByteArrayInputStream	getByteArrayInputStream() This method returns a reference to a byte array input stream representing the encoded message.
byte[]	getMsgCopy() This method returns the encoded message in a byte array.
int	getMsgLength() This method returns the length of the encoded message component.
void	initBuffer(int size)
void	reset() This method resets the buffer to allow a new record to be encoded into it.
java.lang.String	toString() This method will return a string representation of the data in the encode buffer.
void	write(java.io.OutputStream out) This method writes the encoded record to the given output stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mByteIndex

```
protected int mByteIndex
```

mData

```
protected byte mData
```

Constructors

Asn1EncodeByteBufferRev

```
public Asn1EncodeByteBufferRev()
```

The default constructor initializes the encode buffer to the default buffer size (INITIAL_SIZE bytes).

Asn1EncodeByteBufferRev

```
public Asn1EncodeByteBufferRev(int size)
```

This constructor allows specification of the initial size. This overrides the INITIAL_SIZE constant value.

Parameters:

`size` - initial Buffer size

Methods

checkSize

```
protected void checkSize(int bytesRequired)
```

This method determines if the encode buffer can hold the requested number of bytes. If not, the buffer is expanded. This treats mByteIndex as unused and reflects the fact that we're writing into the buffer from right to left.

Parameters:

`bytesRequired` - Number of required bytes.

copy

```
public void copy(byte value)
```

This method copies a single byte to the encode buffer.

Parameters:

`value` - The byte value to copy

copy

```
public void copy(byte[] value)
  throws Asn1Exception
```

This method copies multiple bytes to the encode buffer

(continued from last page)

Parameters:
`value` - Array of bytes to copy to the encode buffer

copy

```
public void copy(byte[] value,
                 int startOffset,
                 int length)
  throws Asn1Exception
```

This method copies multiple bytes to the encode buffer

Parameters:
`value` - Array of bytes to copy to the encode buffer

copy

```
public void copy(java.lang.String value)
```

This method copies a character string into the encode buffer

Parameters:
`value` - String to copy to the encode buffer

getByteArrayInputStream

```
public java.io.ByteArrayInputStream getByteArrayInputStream()
```

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

Returns:
byte array input stream containing encoded message

getMsgCopy

```
public final byte[] getMsgCopy()
```

This method returns 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.

Returns:
byte array containing encoded message

getMsgLength

```
public int getMsgLength()
```

This method returns the length of the encoded message component.

Returns:
length of encoded message component

initBuffer

```
protected void initBuffer(int size)
```

(continued from last page)

reset

```
public void reset()
```

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

toString

```
public java.lang.String toString()
```

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

Returns:

Stringified representation of the value

write

```
public final void write(java.io.OutputStream out)
  throws java.io.IOException
```

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

Parameters:

`out` - Output stream to which record is to be written

com.objsys.asn1j.runtime
Class Asn1EndOfBufferException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1EndOfBufferException
```

public class **Asn1EndOfBufferException**
extends Asn1Exception

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..

Constructor Summary

public	`Asn1EndOfBufferException(Asn1DecodeBuffer buffer)` This constructor creates an exception object with a textual message describing the tag of the duplicate element..

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1EndOfBufferException

`public` **`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

com.objsys.asn1j.runtime
Class Asn1Enumerated

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Enumerated
```

public abstract class **Asn1Enumerated**
extends Asn1Type

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.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 10).
static final int	UNDEFINED The **UNDEFINED**constant is stored in the **value**member variable when the value of this enumerated type is undetermined. Value: **-999**
int	value This public member variable is where the enumerated value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Enumerated() The default constructor sets the enumerated value to undefined.
public	Asn1Enumerated(int value_) This constructor creates an enumerated object from a integer value.

Method Summary

java.lang.Object	clone() Returns a "clone" of this object.

int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 enumerated value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode the enumerated value to JSON.
void	`encode(Asn1OerEncodeBuffer buffer)` Encode enumerated value according to OER.
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This method encodes an ASN.1 enumerated value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out,long lower,long upper)` This method encodes an ASN.1 enumerated value using the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncodeBuffer buffer)` This method encodes an ASN.1 enumerated value using the XML encoding rules (XER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 enumerated value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 enumerated value using the Obj-Sys XML Encoding rules.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix,boolean asText)` This method encodes an ASN.1 enumerated value.
boolean	`equals(int ivalue)` This method compares this enumerated value to the given value for equality.
boolean	`equals(java.lang.Object o)` This method compares this enumerated value to the given value for equality.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
int	`getValue()` This method returns the underlying integer value that is associated with this enumerated object.
int	`hashCode()` This method returns the hash code for the enumerated value.
static int	`parseValue(java.lang.String value)` This method will parse the given enumeration text and return the associated integer value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 10).

UNDEFINED

public static final int **UNDEFINED**

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

value

protected transient int **value**

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.

Constructors

Asn1Enumerated

public **Asn1Enumerated**()

The default constructor sets the enumerated value to undefined.

Asn1Enumerated

public **Asn1Enumerated**(int value_)

This constructor creates an enumerated object from a integer value.

Parameters:

value_ - Integer value

Methods

getAsn1TypeName

public java.lang.String **getAsn1TypeName**()

Returns the ASN.1 type name.

(continued from last page)

Returns:
The ASN.1 type name ENUMERATED.

getValue

```
public final int getValue()
```

This method returns the underlying integer value that is associated with this enumerated object. The integer value is protected to enforce consistency with the static singletons in generated enumerated types.

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:
Length of component or negative status value

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         long lower,
         long upper)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode enumerated value according to OER.

Parameters:
`buffer`

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:
`buffer` - Encode message buffer object
`elemName` - Element name

encode

```
public void encode(Asn1XerEncodeBuffer buffer)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 enumerated value using the Obj-Sys XML Encoding rules. It encodes the value as XML text.

Parameters:

`buffer` - Encode message buffer object
`elemName` - Element name
`nsPrefix` - XML element name space prefix

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix,
         boolean asText)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 enumerated value. It is for use with extended-XER.

Parameters:

`buffer` - Encode message buffer object
`elemName` - Element name
`nsPrefix` - XML element name space prefix
`asText` - If true, encode the value as XML text, otherwise encode as an empty element.

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode the enumerated value to JSON.

Parameters:

`outstream`

equals

```
public boolean equals(int ivalue)
```

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

Parameters:

`ivalue` - Enumerated test value

equals

```
public boolean equals(java.lang.Object o)
```

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

Parameters:

`o` - Asn1Enumerated object to compare to this object

hashCode

```
public int hashCode()
```

This method returns the hash code for the enumerated value.

parseValue

```
public static int parseValue(java.lang.String value)
  throws Asn1Exception
```

This method will parse the given enumeration text and return the associated integer value. This method simply returns -1; it must be overridden by a subclass to be useful. An appropriate override is generated only when generating code for XER or XML encodings.

Returns:

The integer corresponding to the given enumeration text. This can be used with the generated valueOf method to obtain the corresponding instance of the subclass.

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - PER Output Stream object
`lower` - Smallest enumerated value in the set
`upper` - Largest enumerated value in the set

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

clone

```
public java.lang.Object clone()
  throws java.lang.CloneNotSupportedException
```

Returns a "clone" of this object. Because enumerated types are immutable, this implementation returns this object.

com.objsys.asn1j.runtime
Class Asn1Exception

```
java.lang.Object
   |-java.lang.Throwable
      |-java.lang.Exception
         |-java.lang.RuntimeException
            |-com.objsys.asn1j.runtime.Asn1Exception
```

Direct Known Subclasses:
Asn1XmlUnexpElemExc, Asn1XmlMisReqElemExc, Asn1ValueParseException, Asn1UnknownIEI, Asn1UnexpectedElementException, Asn1TagMatchFailedException, Asn1SetDuplicateException, Asn1SeqOrderException, Asn1NotWellFormedXMLException, Asn1NotInSetException, Asn1MissingRequiredException, Asn1MderUnsupported, Asn1InvalidObjectIDException, Asn1InvalidLengthException, Asn1InvalidEnumException, Asn1InvalidElemException, Asn1InvalidChoiceOptionException, Asn1InvalidArgException, Asn1EndOfBufferException, Asn1ConsVioException, Asn1CanonicalException

---

public class **Asn1Exception**
extends java.lang.RuntimeException

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

Constructor Summary

public	`Asn1Exception(java.lang.String message)` This constructor passes the given message text to the superclass.
protected	`Asn1Exception()` This constructor passes the given message text to the superclass.
public	`Asn1Exception(Asn1DecodeBuffer buffer,java.lang.String message)` This constructor creates the base exception object and captures the current buffer offset from the decode buffer..

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1Exception

```
public Asn1Exception(java.lang.String message)
```

This constructor passes the given message text to the superclass.

Parameters:

`message` - Error message text

Asn1Exception

```
protected Asn1Exception()
```

This constructor passes the given message text to the superclass.

Asn1Exception

```
public Asn1Exception(Asn1DecodeBuffer buffer,
                     java.lang.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

com.objsys.asn1j.runtime
Class Asn1GeneralizedTime

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1AbstractTime
               ┗-com.objsys.asn1j.runtime.Asn1GeneralizedTime
```

public class **Asn1GeneralizedTime**
extends Asn1AbstractTime

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 24).

Fields inherited from class com.objsys.asn1j.runtime.Asn1AbstractTime

Apr, April, Aug, August, day, Dec, December, derRules, diffHour, diffMin, Feb, February, hour, Jan, January, Jul, July, Jun, June, Mar, March, May, minute, month, Nov, November, Oct, October, parsed, secFraction, second, Sep, September, utcFlag, year

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString

BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1GeneralizedTime() The default constructor creates an empty time string object.

public	`Asn1GeneralizedTime(boolean useDerRules)` This constructor creates an empty time string object and allows DER encoding rules to be specified.
public	`Asn1GeneralizedTime(java.lang.String data)` This version of the constructor can be used to set the string **value**member variable to the given time string.
public	`Asn1GeneralizedTime(java.lang.String data,boolean useDerRules)` This version of the constructor can be used to set the string **value**member variable to the given time string and specify DER encoding rules be used to construct the string.

Method Summary

boolean	`compileString()` Compiles new time string accoring X.680 (clause 41) and ISO 8601.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
boolean	`equals(java.lang.String s)` This method compares this object with a String value.
int	`getCentury()` This method returns the century part (first two digits) of the year component of the time value.
void	`parseString(java.lang.String string)` This method parses the given time string value.
void	`setCentury(int century)` This method sets the century part (first two digits) of the year component of the time value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1AbstractTime`
`charAt, clear, compareTo, compileString, decode, decode, decodeXML, encode, encode, encode, encode, encode, encodeXER, encodeXMLData, equals, equals, getDay, getDiff, getDiffHour, getDiffMinute, getFraction, getHour, getMinute, getMonth, getSecond, getTime, getUTC, getYear, init, parseInt, parseString, parseXmlString, putInteger, putInteger, safeParseString, setDay, setDER, setDiff, setDiff, setDiffHour, setFraction, setHour, setMinute, setMonth, setSecond, setTime, setUTC, setYear`

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`
`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 24).

Constructors

Asn1GeneralizedTime

```
public Asn1GeneralizedTime()
```

The default constructor creates an empty time string object.

Asn1GeneralizedTime

```
public Asn1GeneralizedTime(boolean 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.

Asn1GeneralizedTime

```
public Asn1GeneralizedTime(java.lang.String data)
```

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

Parameters:

`data` - Character string

(continued from last page)

Asn1GeneralizedTime

```
public Asn1GeneralizedTime(java.lang.String data,
                           boolean useDerRules)
```

This version of the constructor can be used to set the string **value**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.

Methods

equals

```
public boolean equals(java.lang.String s)
```

This method compares this object with a String value. Strictly speaking, the contract of equals stipulates identity: a.equals(b) iff b.equals(a). However, it is never the case that a String will equal an Asn1Time object. In this case, then, equals should be taken to mean that the time represented by the string passed in is the same time as the object to which it is compared.

Parameters:

`s`

Returns:

True, if the input string represents the same time as this object.

---

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
          boolean explicit,
          int implicitLength)
  throws Asn1Exception,
          java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
          boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

(continued from last page)

getCentury

```
public int getCentury()
  throws Asn1Exception
```

This method returns the century part (first two digits) of the year component of the time value.

Returns:

Century part (first two digits) of the year component is returned if the operation is successful.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

setCentury

```
public void setCentury(int century)
  throws Asn1Exception
```

This method sets the century part (first two digits) of the year component of the time value.

Parameters:

`century` - Century part (first two digits) of the year component.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

parseString

```
public void parseString(java.lang.String string)
  throws Asn1Exception
```

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

Parameters:

`string` - The time string value to be parsed.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

compileString

```
protected boolean compileString()
  throws Asn1Exception
```

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

Returns:

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

Throws:

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

(continued from last page)

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

com.objsys.asn1j.runtime
Class Asn1GeneralString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1VarWidthCharString
            ┗-com.objsys.asn1j.runtime.Asn1GeneralString
```

public class **Asn1GeneralString**
extends Asn1VarWidthCharString

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 27).

Fields inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1GeneralString() The default constructor creates an empty string object.
public	Asn1GeneralString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	decode(Asn1JsonDecodeBuffer buffer) Decode ASN.1 GeneralString from JSON.
int	encode(Asn1BerEncodeBuffer buffer,boolean explicit) This method encodes an ASN.1 string type.
void	encode(Asn1BerOutputStream out,boolean explicit) 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.
void	encode(Asn1JsonOutputStream outstream) Encode this GeneralString to JSON.

Methods inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString
decode, decode, decode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 27).

Constructors

(continued from last page)

Asn1GeneralString

`public` **`Asn1GeneralString`**`()`

The default constructor creates an empty string object.

Asn1GeneralString

`public` **`Asn1GeneralString`**`(java.lang.String data)`

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

Parameters:

`data` - Character string

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

(continued from last page)

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 GeneralString from JSON.

Parameters:

`buffer` - The buffer to decode from.

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this GeneralString to JSON.

Parameters:

`outstream` - The stream to encode to.

Throws:

`java.io.IOException` - for I/O exception

com.objsys.asn1j.runtime
Class Asn1GenRtkey

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1GenRtkey
```

public class **Asn1GenRtkey**
extends java.lang.Object

Utility class to create the license java file using license data extracted from osyslic.txt

Method Summary

static void	main(java.lang.String[] args)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

main

```
public static void main(java.lang.String[] args)
```

com.objsys.asn1j.runtime
Class Asn1GraphicString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1VarWidthCharString
            ┗-com.objsys.asn1j.runtime.Asn1GraphicString
```

public class **Asn1GraphicString**
extends Asn1VarWidthCharString

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 25).

Fields inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1GraphicString() The default constructor creates an empty string object.
public	Asn1GraphicString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 GraphicString from JSON.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this GraphicString to JSON.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1VarWidthCharString`
`decode, decode, decode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`
`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 25).

Constructors

(continued from last page)

Asn1GraphicString

`public` **`Asn1GraphicString`**`()`

The default constructor creates an empty string object.

Asn1GraphicString

`public` **`Asn1GraphicString`**`(java.lang.String data)`

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

---

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 GraphicString from JSON.

Parameters:

`buffer` - The buffer to decode from.

Throws:

`java.io.IOException` - for I/O exception

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this GraphicString to JSON.

Parameters:

`outstream` - The stream to encode to.

Throws:

`java.io.IOException` - for I/O exception

com.objsys.asn1j.runtime
Class Asn1IA5String

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1IA5String
```

public class **Asn1IA5String**
extends Asn18BitCharString

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

Field Summary

static final Asn1CharSet	CHARSET
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 22).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1IA5String() The default constructor creates an empty string object.
public	Asn1IA5String(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 IA5 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 IA5 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`

`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

CHARSET

`public static final com.objsys.asn1j.runtime.Asn1CharSet` **`CHARSET`**

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 22).

Constructors

(continued from last page)

Asn1IA5String

```
public Asn1IA5String()
```

The default constructor creates an empty string object.

Asn1IA5String

```
public Asn1IA5String(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Interface Asn1InputStream

All Known Implementing Classes:
Asn1PerInputStream, Asn1MderDecodeBuffer, Asn1DerInputStream, Asn1BerInputStream

---

public interface **Asn1InputStream**

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

Method Summary

abstract 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.
abstract void	`close()` Closes this input stream and releases any system resources associated with the stream.
abstract void	`mark(int readLimit)` This method is used to mark the current position in the input stream for retry processing.
abstract boolean	`markSupported()` Tests if this input stream supports the `mark`and `reset`methods.
abstract void	`reset()` This method is used to reset the current position in the input stream back to the location of the last 'mark' call.
abstract long	`skip(long nbytes)` This method will skip over the requested number of bytes in the input stream.

Methods

available

```
public abstract int available()
  throws java.io.IOException
```

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.

Throws:
`IOException` - if an I/O error occurs.

(continued from last page)

close

```
public abstract void close()
  throws java.io.IOException
```

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

Throws:

`IOException` - if an I/O error occurs.

---

mark

```
public abstract void mark(int readLimit)
```

This method is used to mark the current position in the input stream for retry processing. It is equivalent to the Java InputStream 'mark' method.

Parameters:

`readLimit` - Max number of bytes that can be read before mark becomes invalid.

---

markSupported

```
public abstract boolean markSupported()
```

Tests if this input stream supports the `mark`and `reset`methods. The `markSupported`method of `InputStream`returns `false`.

Returns:

`true`if this true type supports the mark and reset method; `false`otherwise.

---

reset

```
public abstract 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 the Java InputStream 'reset' method.

---

skip

```
public abstract long skip(long nbytes)
  throws java.io.IOException
```

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

Parameters:

`nbytes` - Number of bytes to skip

Returns:

the actual number of bytes skipped.

Throws:

`IOException` - if an I/O error occurs.

com.objsys.asn1j.runtime
Class Asn1Integer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Integer
```

public class **Asn1Integer**
extends Asn1Type

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

Field Summary

static final java.lang.Object	MAX
static final java.lang.Object	MIN
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 2).
long	value This public member variable is where the integer value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Integer() The default constructor sets the integer value to zero.
public	Asn1Integer(long value_) This constructor creates an integer object from a integer value.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified.

void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 INTEGER from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode this value as if unconstrained.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This method decodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,java.lang.Object upper)` This method decodes a semi-constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,java.lang.Object lower,long upper)` This method decodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,java.lang.Object lower,java.lang.Object upper)` This method decodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decode16Bit(Asn1MderDecodeBuffer buffer,boolean signed)` Decode a signed or unsigned 16-bit integer from an MDER encoding.
void	`decode32Bit(Asn1MderDecodeBuffer buffer,boolean signed)` Decode a signed or unsigned 32-bit integer from an MDER encoding.
void	`decode8Bit(Asn1MderDecodeBuffer buffer,boolean signed)` Decode a signed or unsigned 8-bit integer from an MDER encoding.
void	`decodeSigned(Asn1OerDecodeBuffer buffer)` Decode a variable-size signed integer, encoded with a length, into this object, according to OER.
void	`decodeSigned(Asn1OerDecodeBuffer buffer,int octets)` Decode a signed integer (2's complement) of the given length into this object.
void	`decodeTallyMarks(Asn1NasDecodeBuffer buffer,int bit)` Decode this integer's value by decoding "tally marks", where the given bit value (0/1) is used for each tally mark and the negation of the given bit marks the end of the tally marks.
void	`decodeTallyMarksPattern(Asn1NasDecodeBuffer buffer,byte pattern)` Decode this integer's value by decoding "tally marks", where the given bit pattern is used to determine the bit value to use for each tally mark, according to the position within the buffer, and the negation of a tally mark is used to mark the end of them.
void	`decodeUnsigned(Asn1OerDecodeBuffer buffer)` Decode a variable-size unsigned integer, encoded with a length, into this object, according to OER.

void	`decodeUnsigned(Asn1OerDecodeBuffer buffer,int octets)` Decode an unsigned integer (binary integer) of the given length into this object.
static long	`decodeValue(Asn1PerDecodeBuffer buffer)` This method decodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
static long	`decodeValue(Asn1PerDecodeBuffer buffer,long lower,long upper)` This method decodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 integer value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 integer value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 integer value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream out)` Encode the value of this object as a JSON number.
void	`encode(Asn1OerEncodeBuffer buffer)` Encode this value as if unconstrained.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This method encodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,java.lang.Object upper)` This method encodes a semi-constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,java.lang.Object lower,long upper)` This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,java.lang.Object lower,java.lang.Object upper)` This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).

void	`encode(Asn1PerOutputStream out,long lower,long upper)` This method encodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out,long lower,java.lang.Object upper)` This method encodes a semi-constrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out,java.lang.Object lower,long upper)` This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out,java.lang.Object lower,java.lang.Object upper)` This method encodes a unconstrained ASN.1 integer value using the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 integer value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	`encode16Bit(Asn1MderOutputStream out,boolean signed)` This method encodes this ASN.1 INTEGER value to the MDER encoding.
void	`encode32Bit(Asn1MderOutputStream out,boolean signed)` This method encodes this ASN.1 INTEGER value to the MDER encoding.
void	`encode8Bit(Asn1MderOutputStream out,boolean signed)` This method encodes this ASN.1 INTEGER value to the MDER encoding.
void	`encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName)` This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	`encodeBits(Asn1NasEncodeBuffer buffer,int nbits)` Encode the lowest nbits bits of this integer's long value to the given buffer.
void	`encodeMergeBits(Asn1NasEncodeBuffer buffer,int nbits)` Encode the lowest nbits bits of this integer's long value to the given buffer.
void	`encodeSigned(Asn1OerEncodeBuffer buffer)` Encode this value as a variable-size signed integer, with length, according to OER.
void	`encodeSigned(Asn1OerEncodeBuffer buffer,int octets)` Encode integer value as a signed value (2's complement) in the given number of octets.
void	`encodeTallyMarks(Asn1NasEncodeBuffer buffer,int bit)` Encode this integer's value by encoding as many "tally marks", where the given bit value (0/1) is used for each tally mark and the negation of the given bit marks the end of the tally marks.

void	`encodeTallyMarksPattern(Asn1NasEncodeBuffer buffer,byte pattern)` Encode this integer's value by encoding as many "tally marks", where the given bit pattern is used to determine the bit value to use for each tally mark, according to the position within the buffer, and the negation of a tally mark is used to mark the end of them.
void	`encodeUnsigned(Asn1OerEncodeBuffer buffer)` Encode this value as a variable-size unsigned integer, with length, according to OER.
void	`encodeUnsigned(Asn1OerEncodeBuffer buffer,int octets)` Encode integer value as an unsigned value (binary integer) in the given number of octets, according to OER.
static void	`encodeValue(Asn1PerEncoder encoder,long value,long lower,long upper)` This method encodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER).
boolean	`equals(long lvalue)` This method compares this integer value to the given value for equality.
boolean	`equals(java.lang.Object o)` This method compares this integer value to the given value for equality.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
int	`getBitCount()` This method calculates the count of bits in the contained integer value.
static int	`getBitCount(long ivalue)` This method calculates the count of bits in an integer value.
int	`getIntValue()` Return the value of the Asn1Integer as an int and throw an exception if the value does not fit in an int.
int	`getUnsignedBitCount()` This method calculates the count of bits in the contained unsigned integer value.
static int	`getUnsignedBitCount(long ivalue)` This method calculates the count of bits in an unsigned integer value.
int	`hashCode()` This method returns the hash code for this integer value.
java.lang.String	`toString()` This method will return a string representation of the integer value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

MIN

`public static final java.lang.Object` **`MIN`**

MAX

`public static final java.lang.Object` **`MAX`**

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 2).

value

`public transient long` **`value`**

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.

Constructors

Asn1Integer

`public` **`Asn1Integer`**`()`

The default constructor sets the integer value to zero.

Asn1Integer

`public` **`Asn1Integer`**`(long value_)`

This constructor creates an integer object from a integer value.

Parameters:

`value_` - Integer value

Methods

getAsn1TypeName

`public java.lang.String` **`getAsn1TypeName`**`()`

Returns the ASN.1 type name.

Returns:

(continued from last page)

The ASN.1 type name INTEGER.

getIntValue

```
public final int getIntValue()
```

Return the value of the Asn1Integer as an int and throw an exception if the value does not fit in an int.

Returns:

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of component or negative status value

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' in this object.

Parameters:

`buffer` - PER Decode message buffer object

decodeValue

```
public static long decodeValue(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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.

(continued from last page)

Parameters:

`buffer` - PER Decode message buffer object

Returns:

The decoded value

decodeValue

```
public static long decodeValue(Asn1PerDecodeBuffer buffer,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes a constrained ASN.1 integer value using the Packed Encoding Rules (PER).

Parameters:

`buffer` - PER Decode message buffer object
`lower` - Lower bound of the integer range
`upper` - Upper bound of the integer range

Returns:

The decoded result.

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - PER Decode message buffer object
`lower` - Lower bound of the integer range
`upper` - Upper bound of the integer range

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         java.lang.Object lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - PER Decode message buffer object
`lower` - Lower bound equal MIN
`upper` - Upper bound of the integer range

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         long lower,
         java.lang.Object upper)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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 'value' in this object.

Parameters:

`buffer` - PER Decode message buffer object
`lower` - Lower bound of the integer range
`upper` - Upper bound equal MAX

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         java.lang.Object lower,
         java.lang.Object upper)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - PER Decode message buffer object
`lower` - Lower bound equal MIN
`upper` - Upper bound equal MAX

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
         long lower,
         long upper)
  throws Asn1Exception
```

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

---

encodeValue

```
public static void encodeValue(Asn1PerEncoder encoder,
         long value,
         long lower,
         long upper)
  throws Asn1Exception
```

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:

`encoder` - PER Encoder object

(continued from last page)

`value` - The value to encode
`lower` - Lower bound (inclusive) of integer being encoded
`upper` - Upper bound (inclusive) of integer being encoded

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          java.lang.Object lower,
          long upper)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          java.lang.Object upper)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          java.lang.Object lower,
          java.lang.Object upper)
  throws Asn1Exception
```

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

decodeSigned

```
public final void decodeSigned(Asn1OerDecodeBuffer buffer,
          int octets)
  throws java.io.IOException
```

Decode a signed integer (2's complement) of the given length into this object.

Parameters:

`octets` - The number of octets in which the value was encoded. 0 < octets <= 8

(continued from last page)

decodeUnsigned

```
public final void decodeUnsigned(Asn1OerDecodeBuffer buffer,
         int octets)
  throws java.io.IOException
```

Decode an unsigned integer (binary integer) of the given length into this object.

Parameters:

`octets` - The number of octets in which the value was encoded. 0 < octets <= 8

---

decodeSigned

```
public final void decodeSigned(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a variable-size signed integer, encoded with a length, into this object, according to OER.

---

decodeUnsigned

```
public final void decodeUnsigned(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a variable-size unsigned integer, encoded with a length, into this object, according to OER.

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode this value as if unconstrained. Subclasses may override this method to decode according to the constraints on the respective ASN.1 type.

Parameters:

`buffer`

---

encodeSigned

```
public final void encodeSigned(Asn1OerEncodeBuffer buffer)
```

Encode this value as a variable-size signed integer, with length, according to OER.

---

encodeUnsigned

```
public final void encodeUnsigned(Asn1OerEncodeBuffer buffer)
```

Encode this value as a variable-size unsigned integer, with length, according to OER.

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode this value as if unconstrained. Subclasses may override this method to encode it according to the constraints on the respective ASN.1 type.

Parameters:

`buffer`

---

encodeSigned

```
public final void encodeSigned(Asn1OerEncodeBuffer buffer,
         int octets)
```

(continued from last page)

Encode integer value as a signed value (2's complement) in the given number of octets. The value must fit in the given number of octets.

Parameters:

`buffer` - The buffer.
`octets` - The number of octets to encode in: 1, 2, 4, or 8.

---

encodeUnsigned

```
public final void encodeUnsigned(Asn1OerEncodeBuffer buffer,
         int octets)
```

Encode integer value as an unsigned value (binary integer) in the given number of octets, according to OER. The value must be non-negative and fit in the given number of octets.

Parameters:

`buffer` - The buffer.
`octets` - The number of octets to encode in: 1, 2, 4, or 8.

---

decodeTallyMarks

```
public final void decodeTallyMarks(Asn1NasDecodeBuffer buffer,
         int bit)
  throws java.io.IOException
```

Decode this integer's value by decoding "tally marks", where the given bit value (0/1) is used for each tally mark and the negation of the given bit marks the end of the tally marks.

Throws:

`IllegalArgumentException` - if this value is negative

---

decodeTallyMarksPattern

```
public final void decodeTallyMarksPattern(Asn1NasDecodeBuffer buffer,
         byte pattern)
  throws java.io.IOException
```

Decode this integer's value by decoding "tally marks", where the given bit pattern is used to determine the bit value to use for each tally mark, according to the position within the buffer, and the negation of a tally mark is used to mark the end of them. For example, to use "L" bits as the tally mark, use 0x2B for the pattern.

---

encodeTallyMarks

```
public final void encodeTallyMarks(Asn1NasEncodeBuffer buffer,
         int bit)
```

Encode this integer's value by encoding as many "tally marks", where the given bit value (0/1) is used for each tally mark and the negation of the given bit marks the end of the tally marks.

Throws:

`IllegalArgumentException` - if this value is negative

---

encodeTallyMarksPattern

```
public final void encodeTallyMarksPattern(Asn1NasEncodeBuffer buffer,
         byte pattern)
```

Encode this integer's value by encoding as many "tally marks", where the given bit pattern is used to determine the bit value to use for each tally mark, according to the position within the buffer, and the negation of a tally mark is used to mark the end of them. For example, to use "L" bits as the tally mark, use 0x2B for the pattern.

Throws:

`IllegalArgumentException` - if this value is negative

encodeBits

```
public final void encodeBits(Asn1NasEncodeBuffer buffer,
         int nbits)
```

Encode the lowest nbits bits of this integer's long value to the given buffer.

Parameters:

buffer
nbits

encodeMergeBits

```
public final void encodeMergeBits(Asn1NasEncodeBuffer buffer,
         int nbits)
```

Encode the lowest nbits bits of this integer's long value to the given buffer. This will preserve any bits that follow the encoded bits, which is useful when the buffer has been positioned to overwrite some bits in the middle of the buffer.

Parameters:

buffer
nbits

decode8Bit

```
public void decode8Bit(Asn1MderDecodeBuffer buffer,
         boolean signed)
  throws Asn1Exception,
         java.io.IOException
```

Decode a signed or unsigned 8-bit integer from an MDER encoding. This should be used to decode integer types that are constrained to exactly the value space of a signed/unsigned 8 bit integer.

decode16Bit

```
public void decode16Bit(Asn1MderDecodeBuffer buffer,
         boolean signed)
  throws Asn1Exception,
         java.io.IOException
```

Decode a signed or unsigned 16-bit integer from an MDER encoding. This should be used to decode integer types that are constrained to exactly the value space of a signed/unsigned 16 bit integer.

decode32Bit

```
public void decode32Bit(Asn1MderDecodeBuffer buffer,
         boolean signed)
  throws Asn1Exception,
         java.io.IOException
```

Decode a signed or unsigned 32-bit integer from an MDER encoding. This should be used to decode integer types that are constrained to exactly the value space of a signed/unsigned 32 bit integer.

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

buffer - Encode message buffer object

(continued from last page)

`elemName` - Element name

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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` - XML element name space prefix

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

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
`attrName` - Attribute name

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

Parameters:

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

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 INTEGER from JSON.

Parameters:

`buffer`

(continued from last page)

Throws:
`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream out)
  throws java.io.IOException
```

Encode the value of this object as a JSON number.

Parameters:
`out`

equals

```
public boolean equals(long lvalue)
```

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

Parameters:
`lvalue` - Integer test value

equals

```
public boolean equals(java.lang.Object o)
```

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

Parameters:
`o` - Asn1Integer object containing value to compare

hashCode

```
public int hashCode()
```

This method returns the hash code for this integer value.

getBitCount

```
public static int getBitCount(long ivalue)
```

This method calculates the count of bits in an integer value.

Parameters:
`ivalue` - Integer value in which to count bits.

Returns:
Bit count.

getBitCount

```
public int getBitCount()
```

This method calculates the count of bits in the contained integer value.

Returns:
Bit count.

(continued from last page)

getUnsignedBitCount

```
public static int getUnsignedBitCount(long ivalue)
```

This method calculates the count of bits in an unsigned integer value.

Parameters:

`ivalue` - Integer value in which to count bits.

Returns:

Bit count.

---

getUnsignedBitCount

```
public int getUnsignedBitCount()
```

This method calculates the count of bits in the contained unsigned integer value.

Returns:

Bit count.

---

toString

```
public java.lang.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

---

encode

```
public void encode(Asn1BerOutputStream out,
                   boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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:

`out` - PER Encode message buffer object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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:

`out` - PER Encode message buffer object
`lower` - Lower bound (inclusive) of integer being encoded
`upper` - Upper bound (inclusive) of integer being encoded

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
          java.lang.Object lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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:

`out` - PER Encode message buffer object
`lower` - Lower bound equal MIN
`upper` - Upper bound (inclusive) of integer being encoded

encode

```
public void encode(Asn1PerOutputStream out,
          long lower,
          java.lang.Object upper)
  throws Asn1Exception,
          java.io.IOException
```

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:

`out` - PER Encode message buffer object
`lower` - Lower bound (inclusive) of integer being encoded
`upper` - Upper bound equal MAX

encode

```
public void encode(Asn1PerOutputStream out,
          java.lang.Object lower,
          java.lang.Object upper)
  throws Asn1Exception,
          java.io.IOException
```

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.

(continued from last page)

Parameters:

`out` - PER Encode message buffer object
`lower` - Lower bound equal MIN
`upper` - Upper bound equal MAX

encode8Bit

```
public final void encode8Bit(Asn1MderOutputStream out,
         boolean signed)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes this ASN.1 INTEGER value to the MDER encoding. The value must fall in the set of 8-bit unsigned integers (if signed is false) or 8-bit signed integers (if signed is true).

encode16Bit

```
public final void encode16Bit(Asn1MderOutputStream out,
         boolean signed)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes this ASN.1 INTEGER value to the MDER encoding. The value must fall in the set of 16-bit unsigned integers (if signed is false) or 16-bit signed integers (if signed is true).

encode32Bit

```
public final void encode32Bit(Asn1MderOutputStream out,
         boolean signed)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes this ASN.1 INTEGER value to the MDER encoding. The value must fall in the set of 32-bit unsigned integers (if signed is false) or 32-bit signed integers (if signed is true).

com.objsys.asn1j.runtime
Class Asn1InvalidArgException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1InvalidArgException
```

public class **Asn1InvalidArgException**
extends Asn1Exception

This class defines the 'ASN.1 invalid argument' exception that is thrown when as argument that is passed to a method is determined to be invalid (for example, not within a defined range)..

Constructor Summary

public	`Asn1InvalidArgException(java.lang.String argName,java.lang.String argValue)` This constructor creates an exception object with a textual message describing the argument that was invalid..

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1InvalidArgException

```
public Asn1InvalidArgException(java.lang.String argName,
                               java.lang.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

com.objsys.asn1j.runtime
Class Asn1InvalidChoiceOptionException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1InvalidChoiceOptionException
```

public class **Asn1InvalidChoiceOptionException**
extends Asn1Exception

This class defines the 'ASN.1 invalid choice option' exception that is thrown from BER/DER methods when a CHOICE construct is detected to contain an element that is not within the given set.

Constructor Summary

public	`Asn1InvalidChoiceOptionException(Asn1BerDecodeBuffer buffer,Asn1Tag tag)` This constructor creates an exception object with a textual message describing the tag of the invalid element..
public	`Asn1InvalidChoiceOptionException(Asn1PerDecodeBuffer buffer,int index)` This constructor creates an exception object with a textual message describing the PER choice index of the invalid element..
public	`Asn1InvalidChoiceOptionException()` The default constructor is invoked in the encode logic if the object assigned to the choice item is not in the allowed set..

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1InvalidChoiceOptionException

```
public 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

Asn1InvalidChoiceOptionException

```
public 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

Asn1InvalidChoiceOptionException

```
public Asn1InvalidChoiceOptionException()
```

The default constructor is invoked in the encode logic if the object assigned to the choice item is not in the allowed set..

com.objsys.asn1j.runtime
Class Asn1InvalidElemException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1InvalidElemException
```

public class **Asn1InvalidElemException**
extends Asn1Exception

This class defines the 'ASN.1 invalid enum' exception that is thrown when an enumerated value is not within the defined set of values.

Constructor Summary

public	Asn1InvalidElemException(java.lang.String elemName)

Methods inherited from class java.lang.Throwable
addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1InvalidElemException

public **Asn1InvalidElemException**(java.lang.String elemName)

com.objsys.asn1j.runtime
Class Asn1InvalidEnumException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1InvalidEnumException
```

public class **Asn1InvalidEnumException**
extends Asn1Exception

This class defines the 'ASN.1 invalid enum' exception that is thrown when an enumerated value is not within the defined set of values.

Constructor Summary

public	`Asn1InvalidEnumException(long value)` This constructor creates an exception object with a default textual message.
public	`Asn1InvalidEnumException(java.lang.String value)`

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1InvalidEnumException

`public` **`Asn1InvalidEnumException`**`(long value)`

This constructor creates an exception object with a default textual message.

Asn1InvalidEnumException

`public` **`Asn1InvalidEnumException`**`(java.lang.String value)`

com.objsys.asn1j.runtime
Class Asn1InvalidLengthException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1InvalidLengthException
```

public class **Asn1InvalidLengthException**
extends Asn1Exception

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.

Constructor Summary

public	`Asn1InvalidLengthException()` This constructor creates an exception object with a default textual message.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1InvalidLengthException

`public` **`Asn1InvalidLengthException`**`()`

This constructor creates an exception object with a default textual message.

com.objsys.asn1j.runtime
Class Asn1InvalidObjectIDException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1InvalidObjectIDException
```

public class **Asn1InvalidObjectIDException**
extends Asn1Exception

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.

Constructor Summary

public	`Asn1InvalidObjectIDException()` This constructor creates an exception object with a default textual message.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1InvalidObjectIDException

`public` **`Asn1InvalidObjectIDException`**`()`

This constructor creates an exception object with a default textual message.

com.objsys.asn1j.runtime
Class Asn1JsonDecodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1JsonDecodeBuffer
```

public class **Asn1JsonDecodeBuffer**
extends Asn1MessageBufferBase

Decode buffer for decoding JSON-encoded data.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1JsonDecodeBuffer(java.io.Reader reader)` Create Asn1JsonDecodeBuffer on the given reader.
public	`Asn1JsonDecodeBuffer(java.io.InputStream stream)` Create Asn1JsonDecodeBuffer on the given input byte stream.

Method Summary

boolean	`getLazyOpenTypeDecode()` Return true if lazy open type decoding is on.
void	`mark()` Mark the current position so that we can read ahead, without limit, and return to the current position by calling resetPos.
boolean	`nextCharacterIs(char c)` Returns true if the next character is the given character.
boolean	`nextCharacterIsNot(char c)` Returns true if the next character not the given character.
boolean	`readBoolean()` Reads "true" or "false" from input and returns the corresponding boolean value.
void	`readCharacter(char matchChar)` Read the given character from input.
java.lang.String	`readCharStrOctStr()` Read an ASN.1 character string that has been encoded to a JSON string as if it were an ASN.1 octet string.

int	`readInt()` Convenience method equivalent to Integer.parseInt(this.readNumber())
java.lang.String	`readJsonValue()` Read the next JSON value from the input.
void	`readNull()` Reads "null" from input.
java.lang.String	`readNumber()` Read a JSON number from input.
java.lang.String	`readString()` Read a JSON string from the input.
void	`readString(java.lang.String expectedValue)` Convenience method which invokes readString() and throws an exception if the result is not the expectedValue.
void	`reset()` Return the input position to the position at which mark() was previously called.
int	`seekCharacter()` Seeks past whitespace in the input to find the next input character.
void	`setLazyOpenTypeDecode(boolean value)` This method turns lazy open type decoding on or off.
void	`skipJsonValue()` Read past any leading whitespace and the next JSON value from the input.
void	`skipWhitespace()` Skip (read past) all whitespace on input until there is no more input or a nonwhitespace character is the next character of input.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`getContext, hexDump, hexDump, setKey, setTypeCode`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1JsonDecodeBuffer

`public` **`Asn1JsonDecodeBuffer`**`(java.io.Reader reader)`

Create Asn1JsonDecodeBuffer on the given reader.

(continued from last page)

Asn1JsonDecodeBuffer

`public` **`Asn1JsonDecodeBuffer`**`(java.io.InputStream stream)`

Create Asn1JsonDecodeBuffer on the given input byte stream. The constructor will automatically detect the variety of UTF character encoding, following IETF RFC 4627. Note: on Java ME, the input stream is currently assumed to be UTF-8.

Parameters:

`stream`

Methods

getLazyOpenTypeDecode

`public final boolean` **`getLazyOpenTypeDecode`**`()`

Return true if lazy open type decoding is on.

Returns:

true if lazy open type decoding is on

---

skipJsonValue

`public final void` **`skipJsonValue`**`()`
`throws java.io.IOException`

Read past any leading whitespace and the next JSON value from the input. If a JSON value cannot be read from the input, an exception is thrown.

Throws:

`Asn1Exception` - if a JSON value cannot be read.

---

skipWhitespace

`public final void` **`skipWhitespace`**`()`
`throws java.io.IOException`

Skip (read past) all whitespace on input until there is no more input or a nonwhitespace character is the next character of input.

---

mark

`public final void` **`mark`**`()`

Mark the current position so that we can read ahead, without limit, and return to the current position by calling resetPos.

---

nextCharacterIs

`public final boolean` **`nextCharacterIs`**`(char c)`
`throws java.io.IOException`

Returns true if the next character is the given character. This uses seekCharacter to see what the next character is and returns true if and only if there is a next character and it is the given character.

---

nextCharacterIsNot

`public final boolean` **`nextCharacterIsNot`**`(char c)`
`throws java.io.IOException`

Returns true if the next character not the given character. This uses seekCharacter to see what the next character is and returns true if and only if there is a next character and it is not the given character.

(continued from last page)

reset

```
public final void reset()
```

Return the input position to the position at which mark() was previously called.

seekCharacter

```
public final int seekCharacter()
  throws java.io.IOException
```

Seeks past whitespace in the input to find the next input character. This will skip any whitespace in the input until it finds the next input character, call it X. 'X' is not read, so all of the read* methods will still see this character, and repeated calls to seekCharacter will return the same character.

Returns:

The next input character or -1 if there is no next character.

setLazyOpenTypeDecode

```
public final void setLazyOpenTypeDecode(boolean value)
```

This method turns lazy open type decoding on or off. Its setting is relevant only when generating table constraint code (otherwise, open types cannot be decoded). Generated decode methods check this setting to determine whether to decode open types or not.

readCharacter

```
public final void readCharacter(char matchChar)
  throws java.io.IOException
```

Read the given character from input. Leading whitespace is skipped. This will read the next character of input. If it matches the given token, this method simply returns. If there is no next character, or if the next character is not the given character, an exception is thrown.

Parameters:

`matchChar` - The character expected on input.

Throws:

`Asn1Exception` - if the expected character token is not present

readInt

```
public final int readInt()
  throws java.io.IOException
```

Convenience method equivalent to Integer.parseInt(this.readNumber())

readNumber

```
public final java.lang.String readNumber()
  throws java.io.IOException
```

Read a JSON number from input. This will skip any whitespace ahead of the number. If a JSON number is not available in the input, an exception is thrown.

Returns:

A JSON number as a string. It is guaranteed to be a valid JSON number.

Throws:

`IOException`

(continued from last page)

readJsonValue

```
public final java.lang.String readJsonValue()
  throws java.io.IOException
```

Read the next JSON value from the input. Whitespace ahead of the next JSON value is skipped. If a JSON value cannot be read from the input, an exception is thrown. This method will generally be useful in reading an entire JSON object, array, or number. It returns every character of input making up the JSON value (i.e., it does not discard whitespace within the JSON value, or translate escaped characters inside JSON strings). To read a JSON string, or the literals true, false, or null, see readString, readBoolean, and readNull.

Returns:

Throws:

`Asn1Exception` - if a JSON value cannot be read.

---

readString

```
public final void readString(java.lang.String expectedValue)
  throws java.io.IOException
```

Convenience method which invokes readString() and throws an exception if the result is not the expectedValue.

Parameters:

`expectedValue`

Throws:

`IOException`

---

readCharStrOctStr

```
public final java.lang.String readCharStrOctStr()
  throws java.io.IOException
```

Read an ASN.1 character string that has been encoded to a JSON string as if it were an ASN.1 octet string. This is used for TeletexString, T61String, VideotexString, GraphicString, and GeneralString. Whitespace ahead of the string is skipped. If a JSON string is not next on input, an exception is thrown.

Returns:

The character string.

Throws:

`Asn1Exception` - if a JSON string is not on input.

---

readString

```
public final java.lang.String readString()
  throws java.io.IOException
```

Read a JSON string from the input. Whitespace ahead of the string is skipped. If a JSON string is not next on input, an exception is thrown.

Returns:

The JSON string, without the delimiting quote marks and with character escapes applied.

Throws:

`Asn1Exception` - if a JSON string is not on input.

(continued from last page)

readBoolean

```
public final boolean readBoolean()
  throws java.io.IOException
```

Reads "true" or "false" from input and returns the corresponding boolean value. If the input does not match "true" or "false", an exception is thrown. Whitespace ahead of the "true" or "false" value is skipped.

Returns:

Throws:

`Asn1Exception` - if a literal true or false value is not on input.

---

readNull

```
public final void readNull()
  throws java.io.IOException
```

Reads "null" from input. If the input does not match "null", an exception is thrown. Whitespace ahead of the null value is skipped.

Throws:

`Asn1Exception` - if a literal null value is not on input.

com.objsys.asn1j.runtime
Interface Asn1JsonDecoder

public interface **Asn1JsonDecoder**

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

Method Summary

`abstract Asn1Type`	`decode(Asn1JsonDecodeBuffer buffer)` Decode value from given buffer.

Methods

decode

```
public abstract Asn1Type decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode value from given buffer.

Parameters:

`buffer`

Returns:

com.objsys.asn1j.runtime
Class Asn1JsonOutputBuffer

```
java.lang.Object
   |
   +-java.io.Writer
      |
      +-com.objsys.asn1j.runtime.Asn1CharOutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1JsonOutputStream
            |
            +-com.objsys.asn1j.runtime.Asn1JsonOutputBuffer
```

public class **Asn1JsonOutputBuffer**
extends Asn1JsonOutputStream

A byte-array backed Asn1JsonOutputStream that adds a getBytes method.

Fields inherited from class `java.io.Writer`
`lock`

Constructor Summary

public	`Asn1JsonOutputBuffer()` Create an Asn1JsonOutputBuffer that will encode to UTF-8.

Method Summary

`byte[]`	`getBytes()`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1JsonOutputStream`
`encodeAsHexChar, encodeAsHexChars, encodeCharStrOctStr, encodeString`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharOutputStream`
`close, decrLevel, flush, incrLevel, indent, setWriteWhitespace, write`

Methods inherited from class `java.io.Writer`
`append, append, append, close, flush, write, write, write, write, write`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

(continued from last page)

Constructors

Asn1JsonOutputBuffer

`public` **`Asn1JsonOutputBuffer`**`()`

Create an Asn1JsonOutputBuffer that will encode to UTF-8.

Methods

getBytes

`public byte[]` **`getBytes`**`()`

com.objsys.asn1j.runtime
Class Asn1JsonOutputStream

```
java.lang.Object
   |
   +-java.io.Writer
      |
      +-com.objsys.asn1j.runtime.Asn1CharOutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1JsonOutputStream
```

Direct Known Subclasses:
Asn1JsonOutputBuffer

---

public class **Asn1JsonOutputStream**
extends Asn1CharOutputStream

Output stream for JSON Encoding Rules (defined by Obj-Sys). Note that setWriteWhitespace can be used to enable/disable writing of whitespace by the indent method.

Fields inherited from class `java.io.Writer`
`lock`

Constructor Summary

public	`Asn1JsonOutputStream(java.io.Writer writer)` Create a JSON output stream.

Method Summary

void	`encodeAsHexChar(byte value)` Encode the given byte to hexadecimal chars.
void	`encodeAsHexChars(byte[] data,int len)` Encode the given bytes to hexadecimal chars.
void	`encodeCharStrOctStr(java.lang.String value)` Encode the given character string to a JSON string as if the character string were an octet string.
void	`encodeString(java.lang.String value)` Encode the given Java string as a JSON string.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharOutputStream`
`close, decrLevel, flush, incrLevel, indent, setWriteWhitespace, write`

Methods inherited from class `java.io.Writer`
`append, append, append, close, flush, write, write, write, write, write`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1JsonOutputStream

`public` **`Asn1JsonOutputStream`**`(java.io.Writer writer)`

Create a JSON output stream.

Methods

encodeAsHexChar

`public final void` **`encodeAsHexChar`**`(byte value)`
`  throws java.io.IOException`

Encode the given byte to hexadecimal chars.

Parameters:

`value` - The byte to be encoded.

Throws:

`java.io.IOException` - for I/O exception

encodeAsHexChars

`public final void` **`encodeAsHexChars`**`(byte[] data,`
`         int len)`
`  throws java.io.IOException`

Encode the given bytes to hexadecimal chars.

Parameters:

`data` - The bytes to be encoded.
`len` - The number of bytes to encode from data. len <= data.length

Throws:

`java.io.IOException` - for I/O exception

encodeCharStrOctStr

`public final void` **`encodeCharStrOctStr`**`(java.lang.String value)`
`  throws java.io.IOException`

Encode the given character string to a JSON string as if the character string were an octet string. This is used for encoding ASN.1 types TeletexString, T61String, VideotexString, GraphicString, and GeneralString. This will encode the enclosing double quotes.

Parameters:

`value` - The String to encode.

(continued from last page)

encodeString

```
public final void encodeString(java.lang.String value)
  throws java.io.IOException
```

Encode the given Java string as a JSON string. This will encode the enclosing double quotes, and will escape the string's characters as needed.

Parameters:

`value` - The String to encode.

com.objsys.asn1j.runtime
Class Asn1MderDecodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1MderDecodeBuffer
```

All Implemented Interfaces:
Asn1InputStream

---

public class **Asn1MderDecodeBuffer**
extends Asn1DecodeBuffer
implements Asn1InputStream

Asn1MderDecodeBuffer provides the source for an MDER decode.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1MderDecodeBuffer(byte[] msgdata)` Convenience method to create a decoder on an array of data.
public	`Asn1MderDecodeBuffer(java.io.InputStream istream)` Create a decoder on the given input stream.

Method Summary

int	`available()`
void	`close()`
boolean	`markSupported()`
int	`readByte()`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`

addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer

addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase

getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1MderDecodeBuffer

```
public Asn1MderDecodeBuffer(byte[] msgdata)
```

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

Parameters:

msgdata

Asn1MderDecodeBuffer

```
public Asn1MderDecodeBuffer(java.io.InputStream istream)
```

Create a decoder on the given input stream.

Parameters:

istream

Methods

readByte

```
public int readByte()
  throws Asn1Exception,
         java.io.IOException
```

available

```
public int available()
  throws java.io.IOException
```

close

```
public void close()
  throws java.io.IOException
```

markSupported

```
public boolean markSupported()
```

com.objsys.asn1j.runtime
Class Asn1MderOutputStream

```
java.lang.Object
   ┗-java.io.OutputStream
      ┗-com.objsys.asn1j.runtime.Asn1OutputStream
         ┗-com.objsys.asn1j.runtime.Asn1MderOutputStream
```

public class **Asn1MderOutputStream**
extends Asn1OutputStream

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

Fields inherited from class com.objsys.asn1j.runtime.Asn1OutputStream
os

Constructor Summary

public	Asn1MderOutputStream(java.io.OutputStream os)

Methods inherited from class com.objsys.asn1j.runtime.Asn1OutputStream
close, flush, getContext, write, write, write, write2Bytes, write4Bytes

Methods inherited from class java.io.OutputStream
close, flush, write, write, write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1MderOutputStream

public **Asn1MderOutputStream**(java.io.OutputStream os)

com.objsys.asn1j.runtime
Class Asn1MderUnsupported

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1MderUnsupported
```

public class **Asn1MderUnsupported**
extends Asn1Exception

The ASN1C runtime throws this exception whenever it is unable to proceed due to the restrictive nature of the MDER encoding rules. Generally, the conditions under which this exception would be thrown should be detected at the time of ASN.1 compilation, so this exception both suggests a code generation error and an incompatibility between the compiled ASN.1 and the MDER.

Constructor Summary

public	Asn1MderUnsupported()
public	Asn1MderUnsupported(java.lang.String string)

Methods inherited from class java.lang.Throwable
addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1MderUnsupported

public **Asn1MderUnsupported**()

Asn1MderUnsupported

public **Asn1MderUnsupported**(java.lang.String string)

com.objsys.asn1j.runtime
Class Asn1MessageBuffer

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |
      +-com.objsys.asn1j.runtime.Asn1MessageBuffer
```

Direct Known Subclasses:
Asn1EncodeBuffer, Asn1DecodeBuffer

---

public abstract class **Asn1MessageBuffer**
extends Asn1MessageBufferBase

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.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1MessageBuffer()`

Method Summary

void	`addNamedEventHandler(Asn1NamedEventHandler handler)` **Deprecated.** *Use getContext().eventDispatcher.addNamedEventHandler*
int	`getEventHandlerListCount()` **Deprecated.** *use getContext().eventDispatcher.getEventHandlerListCount()*
abstract java.io.InputStream	`getInputStream()` This abstract method must be implemented by all of the derived classes.
boolean	`hasEventHandlers()` **Deprecated.** *use getContext().eventDispatcher.hasEventHandlers()*
void	`invokeCharacters(java.lang.String svalue)` This method is used by the event handling logic to invoke the 'characters' event handling method when message contents are parsed.
void	`invokeEndElement(java.lang.String name,int index)` **Deprecated.** *use getContext().eventDispatcher.endElement(name, index)*
void	`invokeStartElement(java.lang.String name,int index)` **Deprecated.** *use getContext().eventDispatcher.startElement(name, index)*

void	setEventHandlerList(Asn1MessageBuffer buffer) This method will set the event handler dispatcher in this object to be equal to that in the given decode buffer object (the buffers will share the dispatcher).

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1MessageBuffer

public **Asn1MessageBuffer**()

Methods

getInputStream

public abstract java.io.InputStream **getInputStream**()

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

addNamedEventHandler

public void **addNamedEventHandler**(Asn1NamedEventHandler handler)

Deprecated. *Use getContext().eventDispatcher.addNamedEventHandler*

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

Parameters:
handler - Asn1NamedEventHandler object to be added

getEventHandlerListCount

public int **getEventHandlerListCount**()

Deprecated. *use getContext().eventDispatcher.getEventHandlerListCount()*

This method returns number of elements in the event handler list in this object.

Returns:
number of elements in the event handler list.

(continued from last page)

hasEventHandlers

```
public final boolean hasEventHandlers()
```

Deprecated. *use getContext().eventDispatcher.hasEventHandlers()*

This method returns true if, the buffer has event handlers set.

Returns:

true, if the buffer has event handlers. false, otherwise.

---

invokeCharacters

```
public void invokeCharacters(java.lang.String svalue)
```

This method is used by the event handling logic to invoke the 'characters' event handling method when message contents are parsed. It uses the type code last set by `#setTypeCode`

Parameters:

`svalue` - Stringified representation of a parsed value field

---

invokeEndElement

```
public void invokeEndElement(java.lang.String name,
          int index)
```

Deprecated. *use getContext().eventDispatcher.endElement(name, index)*

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

---

invokeStartElement

```
public void invokeStartElement(java.lang.String name,
          int index)
```

Deprecated. *use getContext().eventDispatcher.startElement(name, index)*

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

---

setEventHandlerList

```
public void setEventHandlerList(Asn1MessageBuffer buffer)
```

This method will set the event handler dispatcher in this object to be equal to that in the given decode buffer object (the buffers will share the dispatcher).

Parameters:

`buffer` - Decode buffer object

com.objsys.asn1j.runtime
Class Asn1MessageBufferBase

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1MessageBufferBase
```

Direct Known Subclasses:
Asn1MessageBuffer, Asn1JsonDecodeBuffer

---

public class **Asn1MessageBufferBase**
extends java.lang.Object

Base class for encode and decode message buffers/streams.

Field Summary

Asn1Context	context
short	mTypeCode

Constructor Summary

public	Asn1MessageBufferBase()

Method Summary

Asn1Context	getContext() Return the context object associated with this buffer.
static void	hexDump(java.io.InputStream in) This method prints a formatted hex dump of the contents of the the given input stream to the standard output stream.
static void	hexDump(java.io.InputStream in,java.io.PrintStream out) This method prints a formatted hex dump of the contents of the the given input stream to the given output stream.
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.
void	setTypeCode(short code) This method will sets the internal type code to the given value.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mTypeCode

```
protected short mTypeCode
```

context

```
protected com.objsys.asn1j.runtime.Asn1Context context
```

Constructors

Asn1MessageBufferBase

```
public Asn1MessageBufferBase()
```

Methods

getContext

```
public Asn1Context getContext()
```

Return the context object associated with this buffer.

setKey

```
public 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

setTypeCode

```
public final void setTypeCode(short code)
```

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

Parameters:

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

hexDump

```
public static void hexDump(java.io.InputStream in,
                           java.io.PrintStream out)
```

This method prints a formatted hex dump of the contents of the the given input stream to the given output stream.

(continued from last page)

Parameters:

`in` - InputStream containing data to be dumped
`out` - PrintStream to which formatted data is to be written

hexDump

`public static void` **`hexDump`**`(java.io.InputStream in)`

This method prints a formatted hex dump of the contents of the the given input stream to the standard output stream.

Parameters:

`in` - InputStream containg data to be dumped

com.objsys.asn1j.runtime
Class Asn1MissingRequiredException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1MissingRequiredException
```

public class **Asn1MissingRequiredException**
extends Asn1Exception

This class defines the 'ASN.1 set missing required element' exception that is thrown from BER/DER methods when a SET construct is decoded and found to be missing a required element..

Constructor Summary

public	`Asn1MissingRequiredException(Asn1BerDecodeBuffer buffer)` This constructor creates an exception object with a textual message describing the error.
public	`Asn1MissingRequiredException(Asn1BerDecodeBuffer buffer,java.lang.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 and the location in the buffer where it failed.
public	`Asn1MissingRequiredException(java.lang.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.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1MissingRequiredException

`public` **`Asn1MissingRequiredException`**`(Asn1BerDecodeBuffer buffer)`

(continued from last page)

This constructor creates an exception object with a textual message describing the error.

Parameters:

`buffer` - BER decode buffer object reference

Asn1MissingRequiredException

```
public Asn1MissingRequiredException(Asn1BerDecodeBuffer buffer,
                                    java.lang.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 and the location in the buffer where it failed.

Parameters:

`buffer` - BER decode buffer

`elemName` - Name of missing required element

Asn1MissingRequiredException

```
public Asn1MissingRequiredException(java.lang.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

com.objsys.asn1j.runtime
Class Asn1NamedEventDispatcher

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1NamedEventDispatcher
```

All Implemented Interfaces:
Asn1NamedEventHandler

---

public class **Asn1NamedEventDispatcher**
extends java.lang.Object
implements Asn1NamedEventHandler

Constructor Summary

public	`Asn1NamedEventDispatcher()`

Method Summary

void	`addNamedEventHandler(Asn1NamedEventHandler handler)` This method adds a named event handler to the named event handler list
void	`characters(java.lang.String svalue,short typeCode)`
void	`endElement(java.lang.String name,int index)`
int	`getEventHandlerListCount()` This method returns number of event handlers in the event handler list
boolean	`hasEventHandlers()` This method returns true if there are event handlers to dispatch events to.
void	`startElement(java.lang.String name,int index)`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1NamedEventDispatcher

`public` **`Asn1NamedEventDispatcher`**`()`

Methods

addNamedEventHandler

```
public void addNamedEventHandler(Asn1NamedEventHandler handler)
```

This method adds a named event handler to the named event handler list

Parameters:

`handler` - Asn1NamedEventHandler object to be added

---

getEventHandlerListCount

```
public int getEventHandlerListCount()
```

This method returns number of event handlers in the event handler list

Returns:

number of handlers in the event handler list.

---

hasEventHandlers

```
public final boolean hasEventHandlers()
```

This method returns true if there are event handlers to dispatch events to.

Returns:

true if there are event handlers to dispatch to

---

characters

```
public void characters(java.lang.String svalue,
          short typeCode)
```

---

endElement

```
public void endElement(java.lang.String name,
          int index)
```

---

startElement

```
public void startElement(java.lang.String name,
          int index)
```

com.objsys.asn1j.runtime
Interface Asn1NamedEventHandler

All Known Implementing Classes:
Asn1TraceHandler, Asn1NamedEventDispatcher, Asn1ElementTracker

---

public interface **Asn1NamedEventHandler**

This interface defines the methods that must be implemented to define a SAX-like event handler. These methods are invoked from within the generated Java decode logic when significant events occur during the parsing of an ASN.1 message.

Method Summary

abstract void	`characters(java.lang.String svalue,short typeCode)` The characters callback method is invoked when content (primitive data) is encountered.
abstract void	`endElement(java.lang.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.
abstract void	`startElement(java.lang.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.

Methods

startElement

```
public abstract void startElement(java.lang.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.

---

endElement

```
public abstract void endElement(java.lang.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.

(continued from last page)

characters

```
public abstract void characters(java.lang.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).

com.objsys.asn1j.runtime
Class Asn1NasAltDecodeAction

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1NasAltDecodeAction
```

public class **Asn1NasAltDecodeAction**
extends java.lang.Object

This class represents the actions in Asn1NasAltDecodeTabEntry (which see). The action can be a goto action, indicating the row in the decode table to move to, or an accept action, indicating the choiceId that has been matched.

Field Summary

int	choiceId
int	gotoEntry

Constructor Summary

public	Asn1NasAltDecodeAction(int gotoEntry,int choiceId)

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

gotoEntry

```
public int gotoEntry
```

choiceId

```
public int choiceId
```

Constructors

Asn1NasAltDecodeAction

```
public Asn1NasAltDecodeAction(int gotoEntry,
                              int choiceId)
```

com.objsys.asn1j.runtime
Class Asn1NasAltDecodeTabEntry

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1NasAltDecodeTabEntry
```

public class **Asn1NasAltDecodeTabEntry**
extends java.lang.Object

This class represents a row in a table used to drive decoding of alternative determinants where the determinants are not all of the same length. The table represents a binary tree, with each row representing a node. Each row has a flag indicating whether the next bit decoded should be understood as a zero/one bit or as an L/H bit. Each row also has two actions, one to be followed when a 0/L bit is decoded and the other when a 1/H bit is decoded. Decoding begins at the root (the first entry) and progresses through the tree with each bit read from the input, until the determinant is determined.

Field Summary

boolean	`lhflag` lhflag: true if an L/H bit should be read at this point
Asn1NasAltDecodeAction	`oneHAction` oneHAction: action to take if 1/H decoded
Asn1NasAltDecodeAction	`zeroLAction` zeroLAction: action to take if 0/L decoded

Constructor Summary

public	`Asn1NasAltDecodeTabEntry(boolean lhflag,Asn1NasAltDecodeAction zeroLAction,Asn1NasAltDecodeAction oneHAction)`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

lhflag

`public boolean` **`lhflag`**

lhflag: true if an L/H bit should be read at this point

zeroLAction

`public com.objsys.asn1j.runtime.Asn1NasAltDecodeAction` **`zeroLAction`**

zeroLAction: action to take if 0/L decoded

oneHAction

```
public com.objsys.asn1j.runtime.Asn1NasAltDecodeAction oneHAction
```

oneHAction: action to take if 1/H decoded

Constructors

Asn1NasAltDecodeTabEntry

```
public Asn1NasAltDecodeTabEntry(boolean lhflag,
                                Asn1NasAltDecodeAction zeroLAction,
                                Asn1NasAltDecodeAction oneHAction)
```

com.objsys.asn1j.runtime
Class Asn1NasDecodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1DecodeBitBuffer
               ┗-com.objsys.asn1j.runtime.Asn1NasDecodeBuffer
```

public class **Asn1NasDecodeBuffer**
extends Asn1DecodeBitBuffer

This class handles non-ASN.1 encodings related to 3GPP specifications. NAS stands for Non-Access Stratum, though not everything this class is used for is necessarily part of NAS; we use the term NAS as more of an umbrella term for messages that are not defined using ASN.1.

Field Summary

int	`mVersion` 3GPP Release Version.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`
`mTraceHandler`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1NasDecodeBuffer(byte[] msgdata)` This constructor creates a NAS decode buffer object that references an encoded message.
public	`Asn1NasDecodeBuffer(java.io.InputStream istream)` This constructor creates a NAS decode buffer object that references an encoded message.

Method Summary

boolean	`containerHasMoreBits()` Return true if there are more bits to be read in the current length-constrained container, which is possibly the outer PDU.
int	`decodeAltDetmnt(int detmntBits,int lhmask)` Decode an alternative determinant value in the given number of bits, translating L/H bits according to the pattern 0x2B and based on the buffer's bit offset.
void	`decodeL3NonImperative(boolean callCtrl)` This function decodes the non-imperative part of an L3 message having an empty non-imperative part.
void	`decodeMCCMNC(Asn1IA5String mcc,Asn1IA5String mnc)` Decode an MCC-MNC pair.
java.lang.String	`decodeMobileIdentityType1(int len,boolean odd,byte digit1)` This function is used to decode common data present in different alternatives within the MobileIdentify CHOICE type.
int	`decodeVarAltDetmnt(Asn1NasAltDecodeTabEntry[] table)` Decode a variable length alternative determinant according to the given table.
long	`getContainerRemBits()` Return the number of bits remaining to be read in the current container.
boolean	`getPatternBit(byte pattern)` This function picks out the bit from the given pattern that corresponds to the last bit read from this buffer.
boolean	`isInsideContainer()` Return true if a container length is on the container stack.
void	`popAllContainers()` Pop all containers from the container stack.
void	`popContainer()` Notify the runtime layer of the end of decoding of a length-constrained container of the given length.
void	`pushContainerBits(long bits)` Notify the runtime layer of the start of decoding of a length-constrained container of a given length.
void	`pushContainerBytes(long bytes)` Notify the runtime layer of the start of decoding of a length-constrained container of a given length.
void	`skipToContainerEnd()` Skip all bits, up to and including the last bit of the container from the top of the container stack.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`

binDump, binDump, byteAlign, decodeBit, decodeBitsToInt, decodeBitsToLong, decodeBitsToOctetArray, decodeBitsToOctetArray, getAvailableBits, getBitOffset, getBitOffsetInByte, getMsgBitCnt, getTraceHandler, hasMoreBits, mark, moveBitCursor, readByte, reset, setInputStream, skipBits

Methods inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer
addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mVersion

public int **mVersion**

3GPP Release Version. Setting this to 0 signals decoders to decode data for all 3GPP release versions. Setting this to a different value signals decoders to skip extensions from later versions (but not all decoders will pay attention to this).

Constructors

Asn1NasDecodeBuffer

public **Asn1NasDecodeBuffer**(byte[] msgdata)

This constructor creates a NAS decode buffer object that references an encoded message.

Parameters:

msgdata - Byte array containing an encoded message.

Asn1NasDecodeBuffer

public **Asn1NasDecodeBuffer**(java.io.InputStream istream)

This constructor creates a NAS decode buffer object that references an encoded message. In this case, the message is passed in using an InputStream object.

Parameters:

istream - Input stream containing an encoded message.

(continued from last page)

Methods

containerHasMoreBits

```
public final boolean containerHasMoreBits()
```

Return true if there are more bits to be read in the current length-constrained container, which is possibly the outer PDU. See also pushContainer(Bits|Bytes)/popContainer

---

decodeAltDetmnt

```
public final int decodeAltDetmnt(int detmntBits,
         int lhmask)
  throws java.io.IOException
```

Decode an alternative determinant value in the given number of bits, translating L/H bits according to the pattern 0x2B and based on the buffer's bit offset. The lhmask specifies which bits are L/H bits and which are simply 0/1 bits.

Parameters:

`detmntBits` - The number of bits to decode.

`lhmask` - A mask which indicates whether each bit in detmnt is a 0/1 bit or an L/H bit. L/H bits are represented in detmnt using 0/1, respectively. This is left-aligned so that the highest bit in the mask applies to the first bit to be decoded.

Returns:

The decoded determinant value.

---

decodeMCCMNC

```
public void decodeMCCMNC(Asn1IA5String mcc,
         Asn1IA5String mnc)
  throws java.io.IOException
```

Decode an MCC-MNC pair.

Parameters:

`mcc` - Receives the MCC

`mnc` - Receives the MNC

---

decodeVarAltDetmnt

```
public int decodeVarAltDetmnt(Asn1NasAltDecodeTabEntry[] table)
  throws java.io.IOException
```

Decode a variable length alternative determinant according to the given table.

Parameters:

`table` - Controls decoding. See Asn1NasAltDecodeTabEntry.

Returns:

The decoded alternative.

---

decodeL3NonImperative

```
public void decodeL3NonImperative(boolean callCtrl)
  throws Asn1UnknownIEI,
         java.io.IOException
```

This function decodes the non-imperative part of an L3 message having an empty non-imperative part.

Parameters:

`callCtrl` - Boolean indicating this is a call control message.

(continued from last page)

Throws:

`Asn1UnknownIEI` - if an unknown comprehension required IE is found.

---

decodeMobileIdentityType1

```
public final java.lang.String decodeMobileIdentityType1(int len,
          boolean odd,
          byte digit1)
  throws java.io.IOException
```

This function is used to decode common data present in different alternatives within the MobileIdentify CHOICE type. These include the IMSI, IMEI, and IMEISV alternatives.

Parameters:

`len` - Length in bytes of the data to be decoded. This includes 1 byte for the header.

`odd` - Odd/even indicator boolean decoded from header.

`digit1` - First BCD character decoded from header.

Returns:

decoded value

---

getPatternBit

```
public final boolean getPatternBit(byte pattern)
```

This function picks out the bit from the given pattern that corresponds to the last bit read from this buffer. For example, if the last read bit was the most significant bit of the byte, then this will return the value of the most significant bit in the given pattern. This can be used to get the value of a CSN.1 "L" bit by using pattern 0x2B or an H bit using pattern 0xD4.

Returns:

true for 1 bit, false for 0 bit, as described above

---

getContainerRemBits

```
public final long getContainerRemBits()
```

Return the number of bits remaining to be read in the current container. This can only be called when isInsideContainer() returns true.

Returns:

---

isInsideContainer

```
public final boolean isInsideContainer()
```

Return true if a container length is on the container stack. When this returns true, getContainerBits() returns the number of bits remaining in the container.

Returns:

---

pushContainerBytes

```
public final void pushContainerBytes(long bytes)
```

Notify the runtime layer of the start of decoding of a length-constrained container of a given length. This method should be called when the next bit to be decoded is the first bit of the length-constrained content. This pushes an entry onto the container stack.

Parameters:

`bytes` - Number of bytes of the length-constrained container.

pushContainerBits

```
public final void pushContainerBits(long bits)
```

Notify the runtime layer of the start of decoding of a length-constrained container of a given length. This method should be called when the next bit to be decoded is the first bit of the length-constrained content. This pushes an entry onto the container stack.

Parameters:

`bits` - Number of bits in the length-constrained container.

popContainer

```
public final void popContainer()
```

Notify the runtime layer of the end of decoding of a length-constrained container of the given length. This method should be called when the final bit to be decoded has been decoded. This pops an entry off of the container stack.

popAllContainers

```
public final void popAllContainers()
```

Pop all containers from the container stack. This is useful for clearing the stack when an error has occurred and the decode buffer is being reused.

skipToContainerEnd

```
public final void skipToContainerEnd()
  throws java.io.IOException,
         Asn1EndOfBufferException
```

Skip all bits, up to and including the last bit of the container from the top of the container stack. The container stack must not be empty when this is invoked. This does not pop the container off the container stack.

com.objsys.asn1j.runtime
Interface Asn1NasDecoder

public interface **Asn1NasDecoder**

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

Method Summary

`abstract Asn1Type`	`decode(Asn1NasDecodeBuffer buffer)` Decode value from given buffer.

Methods

decode

```
public abstract Asn1Type decode(Asn1NasDecodeBuffer buffer)
  throws java.io.IOException
```

Decode value from given buffer.

Parameters:

`buffer`

Returns:

com.objsys.asn1j.runtime
Class Asn1NasEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeBitBuffer
               ┗-com.objsys.asn1j.runtime.Asn1NasEncodeBuffer
```

public class **Asn1NasEncodeBuffer**
extends Asn1EncodeBitBuffer

This class handles non-ASN.1 encodings related to 3GPP specifications. NAS stands for Non-Access Stratum, though not everything this class is used for is necessarily part of NAS; we use the term NAS as more of an umbrella term for messages that are not defined using ASN.1.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBitBuffer`
`mByteIndex`, `mData`, `mTraceHandler`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`
`INITIAL_SIZE`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1NasEncodeBuffer()` This constructor creates an encode buffer object with the default initial size.
public	`Asn1NasEncodeBuffer(int size)` This constructor creates an encode buffer object with the given initial size.

Method Summary

void	`encodeAltDetmnt(int detmnt,int detmntBits,int lhmask)` Encode the given alternative determinant value in the given number of bits, translating L/H bits according to the pattern 0x2B and based on the buffer's bit offset.
void	`encodeBitsPattern(byte pattern,int nbits)` This function encodes the given number of bits using a repeating bit pattern.

void	`encodeByteAlignPattern(byte pattern)` This function will byte-align the buffer by encoding according to the given pattern.
void	`encodeMCCMNC(Asn1IA5String mcc,Asn1IA5String mnc)` Encode an MCC-MNC pair.
void	`encodeMobileIdentityType1(int type,java.lang.String str)` Encode alternatives of MobileIdentity.
byte	`getPatternBit(byte pattern)` This function picks out the bit from the given pattern that corresponds to the current bit offset in buffer.
boolean	`isContainerClosed()` Return true if the container closed flag is set.
void	`setContainerClosed(boolean value)` Set or clear the container closed flag.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBitBuffer`

`binDump, byteAlign, checkSize, copy, copy, encodeBit, encodeBits, encodeBits, encodeBits, encodeLongBits, encodeLongBits, getBitOffsetInByte, getBuffer, getByteArrayInputStream, getByteIndex, getMsgBitCnt, getMsgByteCnt, getMsgCopy, getMsgLength, getTraceHandler, hexDump, isByteAligned, reset, reverseBytes, setMsgBitCnt, toString, write`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`

`binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBuffer`

`addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`

`getContext, hexDump, hexDump, setKey, setTypeCode`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1NasEncodeBuffer

```
public Asn1NasEncodeBuffer()
```

This constructor creates an encode buffer object with the default initial size. Whenever the buffer becomes full, it will be expanded.

Asn1NasEncodeBuffer

```
public Asn1NasEncodeBuffer(int size)
```

This constructor creates an encode buffer object with the given initial size. Whenever the buffer becomes full, it will be expanded. For best performance, this size should be large enough to prevent resizing in normal operation.

Parameters:

`size` - The initial size in bytes of an encode buffer.

Methods

encodeAltDetmnt

```
public final void encodeAltDetmnt(int detmnt,
          int detmntBits,
          int lhmask)
```

Encode the given alternative determinant value in the given number of bits, translating L/H bits according to the pattern 0x2B and based on the buffer's bit offset. The lhmask specifies which bits are L/H bits and which are simply 0/1 bits.

Parameters:

`detmnt` - The determinant value to encode (in lowest bits)

`detmntBits` - The number of bits from detmnt to encode.

`lhmask` - A mask which indicates whether each bit in detmnt is a 0/1 bit or an L/H bit. L/H bits are represented in detmnt using 0/1, respectively. This is left-aligned so that the highest bit in the mask applies to the first bit to be encoded.

---

encodeBitsPattern

```
public void encodeBitsPattern(byte pattern,
          int nbits)
```

This function encodes the given number of bits using a repeating bit pattern. This function may be used to encode any number of bits (including more than 8 bits). It will take the bit values to encode from the pattern, in accordance with the buffer's bit offset. For example, if the next bit to encode is the highest bit of the next byte in the buffer, then the bit value encoded will be the highest bit of the pattern.

Parameters:

`pattern` - The repeating pattern.

`nbits` - The number of bits to encode.

---

encodeByteAlignPattern

```
public final void encodeByteAlignPattern(byte pattern)
```

This function will byte-align the buffer by encoding according to the given pattern. If the buffer is not aligned, the lowest bits of the current byte, which have not be written already, will be set to the corresponding lowest bits of the given pattern.

---

encodeMCCMNC

```
public void encodeMCCMNC(Asn1IA5String mcc,
          Asn1IA5String mnc)
  throws java.io.IOException
```

Encode an MCC-MNC pair.

(continued from last page)

Parameters:

`mcc`
`mnc`

---

encodeMobileIdentityType1

```
public final void encodeMobileIdentityType1(int type,
         java.lang.String str)
```

Encode alternatives of MobileIdentity.

Parameters:

`type -` Identifies the alternative used. Values correspond to the values generated for choice for MobileIdentity: IMSI = 1 IMEI = 2 IMEISV = 3 TMSI = 4 TMGI = 5
`str -` The value of the alternative.

---

getPatternBit

```
public final byte getPatternBit(byte pattern)
```

This function picks out the bit from the given pattern that corresponds to the current bit offset in buffer. For example, if the buffer's bit offset is set to write to the most significant next, then this will return the value of the most significant bit in the given pattern. This can be used to get the value of a CSN.1 "L" bit by using pattern 0x2B or an H bit using pattern 0xD4.

Returns:

bit value of 0 or 1 as described above

---

isContainerClosed

```
public final boolean isContainerClosed()
```

Return true if the container closed flag is set.

Returns:

---

setContainerClosed

```
public final void setContainerClosed(boolean value)
```

Set or clear the container closed flag.

Parameters:

`value`

com.objsys.asn1j.runtime
Class Asn1NasUtil

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1NasUtil
```

public class **Asn1NasUtil**
extends java.lang.Object

Field Summary

static final int	FREQ_LIST_FMT_BITMAP0 Enumeration of the formats supported for ARFCN encoding. Value: **0**
static final int	FREQ_LIST_FMT_RANGE_1024 Value: **1**
static final int	FREQ_LIST_FMT_RANGE_128 Value: **4**
static final int	FREQ_LIST_FMT_RANGE_256 Value: **3**
static final int	FREQ_LIST_FMT_RANGE_512 Value: **2**
static final int	FREQ_LIST_FMT_VAR_BITMAP Value: **5**

Constructor Summary

public	Asn1NasUtil()

Method Summary

static int[]	decodeARFCN(int format,byte[] data,int numbits) Decode a list of ARFCN (frequency list) that have been encoded as described in 44.018 10.5.2.13 Frequency List and Annex J.
static void	encodeARFCN(int[] arfcns,int format,int range,int f0,Asn1BitString data) Encode a list of ARFCN (frequency list).

static Asn1NasUtil.ARFCNRange	getARFCNRange(int[] arfcns) Determines the minimum range for the given list of ARFCNs according to 44.018 Annex F.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

FREQ_LIST_FMT_BITMAP0

public static final int **FREQ_LIST_FMT_BITMAP0**

Enumeration of the formats supported for ARFCN encoding. These correspond to the formats described in 44.018 for the Frequency List IE and in Appendix J.

FREQ_LIST_FMT_RANGE_1024

public static final int **FREQ_LIST_FMT_RANGE_1024**

FREQ_LIST_FMT_RANGE_512

public static final int **FREQ_LIST_FMT_RANGE_512**

FREQ_LIST_FMT_RANGE_256

public static final int **FREQ_LIST_FMT_RANGE_256**

FREQ_LIST_FMT_RANGE_128

public static final int **FREQ_LIST_FMT_RANGE_128**

FREQ_LIST_FMT_VAR_BITMAP

public static final int **FREQ_LIST_FMT_VAR_BITMAP**

Constructors

Asn1NasUtil

public **Asn1NasUtil**()

Methods

(continued from last page)

decodeARFCN

```
public static int[] decodeARFCN(int format,
         byte[] data,
         int numbits)
```

Decode a list of ARFCN (frequency list) that have been encoded as described in 44.018 10.5.2.13 Frequency List and Annex J.

Parameters:

`format` - The format in which the ARFCNs have been encoded. One of the FREQ_LIST_FMT_* constants.

`data` - The encoding of the ARFCNs. This only includes the bits that actually encode the ARFCNs and not the bits that indicate the choice of the format. For example, in the Frequency List IE, format range 1024, data begins with bit 3 of octet 3.

`numbits` - The number of bits to be decoded.

Returns:

A a newly allocated array, holding the ARFCN values.

---

encodeARFCN

```
public static void encodeARFCN(int[] arfcns,
         int format,
         int range,
         int f0,
         Asn1BitString data)
```

Encode a list of ARFCN (frequency list). The given ARFCNs will be encoded as described in 44.018 10.5.2.13 Frequency List and Annex J. The encoding of the ARFCNs is set into the given Asn1BitString, data. If data's byte array is too small, a new array of the correct size is created. In some cases, ARFCNs are encoded into a fixed number of bits (e.g. Cell Channel Description). In such cases, size the array to that size before calling this method, and then set numbits to the fixed size after calling this method. If necessary, ensure the extra bits are zeroed. Let usedOctets be the actual number of octets used. Then data.numbits will be set to the following value, depending on the format: format value of data.numbits ---------- ---------- ---- bitmap0 usedOctets * 8 - 4 range 1024 usedOctets * 8 - 5 range 512 usedOctets * 8 - 7 range 256 usedOctets * 8 - 7 range 128 usedOctets * 8 - 7 var bitmap usedOctets * 8 - 7 These values of numbits are designed so that the resulting bit string will align to a byte boundary given the known offset that the string will begin at in the overall encoding of the containing IE (e.g. to make the entire Frequency List IE be a multiple of 8 bits). Any of the numbits bits that are not actually needed to encode the given list of ARFCNs in the given format will be set to 0.

Parameters:

`arfcns` - Array of ARFCNs to be encoded. The values must be distinct. Their ordering does not matter.

`format` - The format to apply to the encoding. One of the FREQ_LIST_FMT_* constants. It is an error if the given ARFCNs cannot be encoded in the given format. For bitmap0, all ARFCNs must be in the range 1..124. For range formats 512, 256, and 128, the ARFCNs have a corresponding minimum range of less than or equal to 512, 256, and 128, respectively.

`range` - The minimum range of the given arfcns. The minimum range R is the smallest value such that there is some value in the set, F0, such that every value in the set can be expressed as: (F0 + N) % 1024 for some N in [0, R-1] getARFCNRange can be used to obtain the minimum range and a value for F0. This argument is not used for format bitmap0.

`f0` - An ARFCN from the list serving as F0 for the given range. This argument is not used for format bitmap0.

`data` - Receives the encoded ARFCNs (includes the F0/orig_arfcn fields.)

---

getARFCNRange

```
public static Asn1NasUtil.ARFCNRange getARFCNRange(int[] arfcns)
```

Determines the minimum range for the given list of ARFCNs according to 44.018 Annex F. Each ARFCN must be unique and less than 1024. The range of an empty list of ARFCNs is 0.

Parameters:

`arfcns` - array of ARFCNs

Returns:

Returns an ARFCNRange with: f0, and ARFCN from the array to use as F0. range, The minimum range. A value chosen such that all ARFCNs in the array satisfy: arfcn = (F0 + N) mod 1024 for some N in [0, range-1]

com.objsys.asn1j.runtime
Class Asn1NasUtil.ARFCNRange

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1NasUtil.ARFCNRange
```

public static class **Asn1NasUtil.ARFCNRange**
extends java.lang.Object

Field Summary

int	f0
int	range

Constructor Summary

public	ARFCNRange()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

range

```
public int range
```

f0

```
public int f0
```

Constructors

ARFCNRange

```
public ARFCNRange()
```

com.objsys.asn1j.runtime
Class Asn1NotInSetException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1NotInSetException
```

public class **Asn1NotInSetException**
extends Asn1Exception

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

Constructor Summary

public	`Asn1NotInSetException(Asn1BerDecodeBuffer buffer,Asn1Tag tag)` This constructor creates an exception object with a textual message describing the tag of the duplicate element.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1NotInSetException

```
public Asn1NotInSetException(Asn1BerDecodeBuffer buffer,
                             Asn1Tag tag)
```

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

Parameters:

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

com.objsys.asn1j.runtime
Class Asn1NotWellFormedXMLException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1NotWellFormedXMLException
```

public class **Asn1NotWellFormedXMLException**
extends Asn1Exception

This class defines the 'ASN.1 not well-formed XML' exception that is thrown when well-formed XML text is expected and it is not in this form. This is the case for the value used in the Asn1XmlAnyElem class.

Constructor Summary

public	`Asn1NotWellFormedXMLException(java.lang.String reason)` This constructor creates an exception object with a textual message describing the argument that was invalid..

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1NotWellFormedXMLException

`public` **`Asn1NotWellFormedXMLException`**`(java.lang.String reason)`

This constructor creates an exception object with a textual message describing the argument that was invalid..

Parameters:

`reason` - Text explaining reason XML is considered to be not well-formed.

com.objsys.asn1j.runtime
Class Asn1Null

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Null
```

public class **Asn1Null**
extends Asn1Type

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

Field Summary

static final Asn1Null	NULL_VALUE The **NULL_VALUE**constant represents a NULL value.
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 5).

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Null()

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 null value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	decode(Asn1JsonDecodeBuffer buffer) Decode ASN.1 NULL from JSON.
void	decode(Asn1NasDecodeBuffer buffer)
void	decode(Asn1OerDecodeBuffer buffer) Decode ASN.1 NULL type, using Octet Encoding Rules (OER).

void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 null value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 null value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 null value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this ASN.1 null value to JSON
void	`encode(Asn1NasEncodeBuffer buffer)`
void	`encode(Asn1OerEncodeBuffer buffer)` Encode ASN.1 NULL type, using Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 null value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 null value using the XML Encoding as specified in the XML schema standard(asn2xsd).
boolean	`equals(java.lang.Object o)` Tests equality.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
java.lang.String	`toString()` This method will return a string representation of the null value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 5).

NULL_VALUE

public static final com.objsys.asn1j.runtime.Asn1Null **NULL_VALUE**

The **NULL_VALUE**constant represents a NULL value.

Constructors

Asn1Null

public **Asn1Null**()

Methods

equals

public boolean **equals**(java.lang.Object o)

Tests equality. All NULL values are equal to each other.

getAsn1TypeName

public java.lang.String **getAsn1TypeName**()

Returns the ASN.1 type name.

Returns:

The ASN.1 type name NULL.

(continued from last page)

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length (in octets) of encoded component

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1NasDecodeBuffer buffer)
  throws java.io.IOException
```

(continued from last page)

encode

```
public void encode(Asn1NasEncodeBuffer buffer)
  throws java.io.IOException
```

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 NULL type, using Octet Encoding Rules (OER).

Parameters:

`buffer`

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode ASN.1 NULL type, using Octet Encoding Rules (OER).

Parameters:

`buffer`

---

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

(continued from last page)

`elemName` - Element name
`nsPrefix` - Element namespace prefix

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 null value using the XML schema encoding rules.

Parameters:

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

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 NULL from JSON.

Parameters:

`buffer`

Throws:

`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this ASN.1 null value to JSON

toString

```
public java.lang.String toString()
```

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

Returns:

Stringified representation of the value

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 null value in accordance with the Packed Encoding Rules (PER).

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1NumericString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1NumericString
```

public class **Asn1NumericString**
extends Asn18BitCharString

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

Field Summary

static final Asn1CharSet	CHARSET The **CHARSET**constant specifies the character set for a numeric string in canonical order.
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 18).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString

BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1NumericString() The default constructor creates an empty string object.
public	Asn1NumericString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`

`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

(continued from last page)

Fields

CHARSET

```
public static final com.objsys.asn1j.runtime.Asn1CharSet CHARSET
```

The **CHARSET**constant specifies the character set for a numeric string in canonical order.

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 18).

Constructors

Asn1NumericString

```
public Asn1NumericString()
```

The default constructor creates an empty string object.

Asn1NumericString

```
public Asn1NumericString(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
                   java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1ObjectDescriptor

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1VarWidthCharString
            ┗-com.objsys.asn1j.runtime.Asn1ObjectDescriptor
```

public class **Asn1ObjectDescriptor**
extends Asn1VarWidthCharString

This is a container class for holding the components of an ASN.1 object descriptor value.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 7).

Fields inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString

BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1ObjectDescriptor() The default constructor creates an empty string object.
public	Asn1ObjectDescriptor(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	encode(Asn1BerEncodeBuffer buffer,boolean explicit) This method encodes an ASN.1 string type.
void	encode(Asn1BerOutputStream out,boolean explicit) 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.

Methods inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString
decode, decode, decode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 7).

Constructors

Asn1ObjectDescriptor

public **Asn1ObjectDescriptor**()

The default constructor creates an empty string object.

Asn1ObjectDescriptor

public **Asn1ObjectDescriptor**(java.lang.String data)

(continued from last page)

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

---

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1ObjectIdentifier

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Type
      |
      +-com.objsys.asn1j.runtime.Asn1ObjectIdentifier
```

Direct Known Subclasses:
Asn1RelativeOID

---

public class **Asn1ObjectIdentifier**
extends Asn1Type

This is a container class for holding the components of an ASN.1 object identifier value.

Field Summary

static final int	MAXSUBIDS The **MAXSUBIDS**constant specifies the maximum number of subidentifiers that can appear in an OID value (128). Value: **128**
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 6).
int[]	value The **value**public member variable is where the object identifier value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1ObjectIdentifier() This constructor creates an empty object identifier that can be used in a decode method call to receive an OID value.
public	Asn1ObjectIdentifier(int[] value_) This constructor initializes the object identifier from the given array of integer subidentifier values.

Method Summary

void	`append(int[] value2)` This method appends an object identifier value onto the existing value.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 OBJECT-IDENTIFIER from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode an ASN.1 OBJECT IDENTIFIER that was encoded according to the Octet Encoding Rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 object identifier value using the packed encoding rules (PER).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 object identifier value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 object identifier value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this object identifier to JSON.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 OBJECT IDENTIFIER according to Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 object identifier value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 object identifier value using the packed encoding rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 object identifier value using the XML encoding rules (XER).

void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an ASN.1 object identifier value using the XML Encoding as specified in the XML schema standard(asn2xsd).
boolean	equals(java.lang.Object oid_) This method compares this object identifier to the given one for equality.
java.lang.String	getAsn1TypeName() Returns the ASN.1 type name.
int	hashCode() This method returns the hash code for the Asn1ObjectIdentifier.
java.lang.String	toString() This method will return a string representation of the OID value.
java.lang.String	toXMLValue() Return the XML value representation (defined in X.680) for this object identifier.
void	validate() Do some minimal validation.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 6).

MAXSUBIDS

public static final int **MAXSUBIDS**

The **MAXSUBIDS**constant specifies the maximum number of subidentifiers that can appear in an OID value (128).

value

public transient int **value**

The **value**public member variable is where the object identifier value is stored.

Constructors

Asn1ObjectIdentifier

```
public Asn1ObjectIdentifier()
```

This constructor creates an empty object identifier that can be used in a decode method call to receive an OID value.

Asn1ObjectIdentifier

```
public Asn1ObjectIdentifier(int[] value_)
```

This constructor initializes the object identifier from the given array of integer subidentifier values.

Parameters:

`value_` - Array of subidentifiers

Methods

getAsn1TypeName

```
public java.lang.String getAsn1TypeName()
```

Returns the ASN.1 type name.

Returns:

The ASN.1 type name OBJECT IDENTIFIER.

append

```
public void append(int[] value2)
```

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

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
                   java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

(continued from last page)

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of encoded component in octets

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 object identifier value using the packed encoding rules (PER).

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 object identifier value using the packed encoding rules (PER). The value to be encoded is stored in the `value`public member variable within this class.

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode an ASN.1 OBJECT IDENTIFIER that was encoded according to the Octet Encoding Rules (OER).

Parameters:

`buffer`

Throws:

`java.io.IOException`

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 OBJECT IDENTIFIER according to Octet Encoding Rules (OER).

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

(continued from last page)

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

Parameters:

`buffer` - Encode message buffer object
`elemName` - Element name

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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` - XML element name space prefix

---

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 object identifier value using the XML schema encoding rules.

Parameters:

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

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 OBJECT-IDENTIFIER from JSON.

Parameters:

`buffer`

Throws:

`java.io.IOException`

(continued from last page)

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this object identifier to JSON.

---

equals

```
public boolean equals(java.lang.Object oid_)
```

This method compares this object identifier to the given one for equality.

Parameters:

`oid_` - Object identifier value

---

hashCode

```
public int hashCode()
```

This method returns the hash code for the Asn1ObjectIdentifier.

---

validate

```
protected void validate()
```

Do some minimal validation. Subclasses may override this to adjust for their validation rules (e.g. Asn1RelativeOID overrides this to be more lax). Minimally, the implementation should enforce that value is not null and that there is at least one arc.

---

toXMLValue

```
public java.lang.String toXMLValue()
```

Return the XML value representation (defined in X.680) for this object identifier.

Throws:

`Asn1InvalidObjectIDException` - if this object does not represent a valid object identifier (only some basic validation is done).

---

toString

```
public java.lang.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

---

encode

```
public void encode(Asn1BerOutputStream out,
        boolean explicit)
  throws Asn1Exception,
        java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

(continued from last page)

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 object identifier value using the packed encoding rules (PER). The value to be encoded is stored in the `value`public member variable within this class.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1OctetString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
```

All Implemented Interfaces:
java.lang.Comparable, java.lang.Cloneable, java.io.Serializable, Asn1TypeIF

Direct Known Subclasses:
Asn1TBCDString, Asn1OpenType, Asn1BCDString, Asn1Base64BinaryString

---

public class **Asn1OctetString**
extends Asn1Type
implements Asn1TypeIF, java.io.Serializable, java.lang.Cloneable, java.lang.Comparable

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 4).
byte[]	value This variable holds the octet string value.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1OctetString() This constructor creates an empty octet string that can be used in a decode method call to receive an octet string value.
public	Asn1OctetString(byte[] data) This constructor initializes an octet string from the given byte array.
public	Asn1OctetString(byte[] data,int offset,int nbytes) This constructor initializes an octet string from a portion of the given byte array.
public	Asn1OctetString(java.lang.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.

Method Summary

int	`compareTo(java.lang.Object octstr)` This method compares two Asn1OctetString objects for equality.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 octet string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 octet string from JSON.
void	`decode(Asn1MderDecodeBuffer buffer)` Decode an unconstrained octet string from the MDER encoding into this object.
void	`decode(Asn1MderDecodeBuffer buffer,int constrainedLength)` Decode an octet string from the MDER encoding into this object.
void	`decode(Asn1NasDecodeBuffer buffer,int nbytes)` Decode the given number of bytes from the buffer.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 octet string using the Octet Encoding Rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 octet string value using the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This method decodes a sized ASN.1 octet string value using the packed encoding rules (PER).
void	`decodeAsBase64(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 octet string, encoded in base64 form, from JSON.
void	`decodeAsHex(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 octet string, encoded in standard, hexadecimal form, from JSON.
void	`decodeContent(Asn1OerDecodeBuffer buffer,int numOctets)` This method decodes an ASN.1 OCTET STRING's content that was encoded according to OER, given the total number of octets.
void	`decodeRemainingBits(Asn1NasDecodeBuffer buffer)` Decode all remaining bits in the current container from the buffer.
void	`decodeXER(java.lang.String buffer,java.lang.String attrs,boolean base64)` This method decodes ASN.1 octet string type using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 octet string type using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 octet string value including the UNIVERSAL tag value and length if explicit tagging is specified.

void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this octet string to JSON.
void	`encode(Asn1MderOutputStream out)` Encode this octet string into the MDER encoding.
void	`encode(Asn1MderOutputStream out,int constrainedLength)` Encode this octet string into the MDER encoding.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes this ASN.1 OCTET STRING value, according to OER.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an unconstrained ASN.1 octet string value using the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This method encodes a size-constrained ASN.1 octet string value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an unconstrained ASN.1 octet string value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,long lower,long upper)` This method encodes a size-constrained ASN.1 octet string value using the packed encoding rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes ASN.1 octet string type using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes ASN.1 octet string type as an xmlhstring.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix,boolean base64)` This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	`encodeAsBase64(Asn1JsonOutputStream outstream)` Encode this octet string to JSON using BASE64 form (as required when a BASE64 encoding instruction applies).
void	`encodeAsHex(Asn1JsonOutputStream outstream)` Encode this octet string to JSON using the standard (hexadecimal) form.
void	`encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName)` This method encodes ASN.1 octet string type using the XML Encoding as specified in the XML schema standard(asn2xsd).

static java.lang.String	`encodeBase64Binary(byte[] data)` Encodes bytes into base64 ASCII characters using the algorithm defined in RFC2045, as defined in w3c stadard http://www.w3.org/tr/2001/rec-xmlschema-2-20010502#base64Binary
void	`encodeContent(Asn1OerEncodeBuffer buffer)` This method encodes the content of an ASN.1 OCTET STRING, according to OER.
boolean	`equals(byte[] value)` This method compares this octet string value to the given value for equality.
boolean	`equals(java.lang.Object os)` This method compares this octet string value to the given value for equality.
boolean	`equals(java.lang.String s)` This method compares the given ASN.1 OCTET STRING value to this OCTET STRING value for equality.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
int	`getLength()` This method will return the length of the OCTET STRING in octets.
int	`getMderLength()` Return the length of the MDER encoding for this type, assuming that the type is NOT a fixed length OCTET STRING (if it is fixed length, you need not call this method!).
int	`hashCode()` This method returns the hashcode for the Asn1OctetString.
java.io.InputStream	`toInputStream()` This method will return a byte array input stream representation of the octet string value.
java.lang.String	`toString()` This method will return a string representation of the octet string value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

(continued from last page)

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag TAG`

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 4).

---

value

`public transient byte value`

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.

Constructors

Asn1OctetString

`public Asn1OctetString()`

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

Asn1OctetString

`public Asn1OctetString(byte[] data)`

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

Parameters:

`data` - Byte array containing an octet string in binary form.

Asn1OctetString

```
public Asn1OctetString(byte[] data,
                       int offset,
                       int nbytes)
```

This constructor initializes an octet string from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

Parameters:

`data` - Byte array containing an octet string in binary form.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array

Asn1OctetString

`public Asn1OctetString(java.lang.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

Methods

getAsn1TypeName

`public java.lang.String getAsn1TypeName()`

Returns the ASN.1 type name.

(continued from last page)

Returns:
The ASN.1 type name OCTET STRING.

compareTo

```
public int compareTo(java.lang.Object octstr)
```

This method compares two Asn1OctetString 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

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:
Length of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

(continued from last page)

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
          java.io.IOException
```

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 'value' public member variable within this class.

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 'value' 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

Throws:

`java.io.IOException` - for I/O exception

---

decode

```
public final void decode(Asn1NasDecodeBuffer buffer,
          int nbytes)
  throws java.io.IOException
```

Decode the given number of bytes from the buffer.

Parameters:

`buffer` - The buffer to decode from.
`nbytes` - The number of bytes to decode.

Throws:

`java.io.IOException` - for I/O exception

decodeRemainingBits

```
public final void decodeRemainingBits(Asn1NasDecodeBuffer buffer)
  throws java.io.IOException
```

Decode all remaining bits in the current container from the buffer. Containers are set by pushing/popping container lengths on/off the buffer's container stack. If no containers are on the stack, this decodes all remaining bits from the buffer.

Parameters:

`buffer` - The buffer to decode from.

Throws:

`java.io.IOException` - for I/O exception

decodeContent

```
public final void decodeContent(Asn1OerDecodeBuffer buffer,
         int numOctets)
  throws java.io.IOException
```

This method decodes an ASN.1 OCTET STRING's content that was encoded according to OER, given the total number of octets. This does not decode a length determinant.

Parameters:

`buffer` - Decode message buffer object

`numOctets` - Total number of octets encoding the content.

Throws:

`java.io.IOException` - for I/O exception

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 octet string using the Octet Encoding Rules (OER). This implementation expects a length determinant in the encoding. This method may be overridden for ASN.1 types that have a fixed length to invoke decodeContent with the predetermined length.

encodeContent

```
public final void encodeContent(Asn1OerEncodeBuffer buffer)
```

This method encodes the content of an ASN.1 OCTET STRING, according to OER. (The length determinant is not encoded.)

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes this ASN.1 OCTET STRING value, according to OER. This encodes the length determinant, assuming that the OCTET STRING is not fixed-size constrained. Subclasses may override this to simply call encodeContent for fixed-size constrained strings.

Parameters:

`buffer` - Encode message buffer object

Throws:

(continued from last page)

`java.io.IOException` - for I/O exception

decode

```
public void decode(Asn1MderDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

Decode an unconstrained octet string from the MDER encoding into this object.

Throws:

`java.io.IOException` - for I/O exception

decode

```
public void decode(Asn1MderDecodeBuffer buffer,
         int constrainedLength)
  throws Asn1Exception,
         java.io.IOException
```

Decode an octet string from the MDER encoding into this object.

Parameters:

`buffer` - The buffer to decode from.

`constrainedLength` - The constrained length of the type being encoded. Pass -1 if the type is unconstrained. For a constrained length octet string, exactly that many octets will be read.

Throws:

`java.io.IOException` - for I/O exception

getMderLength

```
public int getMderLength()
```

Return the length of the MDER encoding for this type, assuming that the type is NOT a fixed length OCTET STRING (if it is fixed length, you need not call this method!).

Returns:

the length

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs,
         boolean base64)
  throws Asn1Exception
```

This method decodes ASN.1 octet string type using the XML encoding rules (XER). Extended-XER is supported by the base64 parameter.

Parameters:

(continued from last page)

`buffer` - String containing data to be decoded
`attrs` - Attributes string from element tag
`base64` - pass true if encoding is base64 (extended-XER only)

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes ASN.1 octet string type as an xmlhstring.

Parameters:

`buffer` - Encode message buffer object
`elemName` - XML element name used to wrap string
`nsPrefix` - XML element name space prefix

Throws:

`java.io.IOException` - for I/O exception

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix,
         boolean base64)
  throws java.io.IOException,
         Asn1Exception
```

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` - XML element name space prefix
`base64` - Pass true to encode as base64 (extended-XER only, including XSD compilation)

Throws:

`java.io.IOException` - for I/O exception

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

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
`attrName` - XML element name used to wrap string

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 octet string type using the XML schema encoding rules.

(continued from last page)

Parameters:
`buffer` - String containing data to be decoded
`attrs` - Attributes string from element tag

decodeAsBase64

```
public final void decodeAsBase64(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 octet string, encoded in base64 form, from JSON.

Parameters:
`buffer` - The buffer to decode from.

Throws:
`java.io.IOException` - for I/O exception

decodeAsHex

```
public final void decodeAsHex(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 octet string, encoded in standard, hexadecimal form, from JSON.

Parameters:
`buffer` - The buffer to decode from.

Throws:
`java.io.IOException` - for I/O exception

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 octet string from JSON. This class's implementation decodes using decodeAsHex, but subclasses may override this to use decodeAsBase64 if encoding instructions require it.

Parameters:
`buffer` - The buffer to decode from.

Throws:
`java.io.IOException` - for I/O exception

encodeAsBase64

```
public final void encodeAsBase64(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this octet string to JSON using BASE64 form (as required when a BASE64 encoding instruction applies).

encodeAsHex

```
public final void encodeAsHex(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this octet string to JSON using the standard (hexadecimal) form.

Parameters:
`outstream` - The stream to encode to.

(continued from last page)

Throws:

`java.io.IOException -` for I/O exception

---

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this octet string to JSON. The implementation for this class uses encodeAsHex; subclasses may override this to use encodeAsBase64 if encoding instructions require it.

Parameters:

`outstream -` The stream to encode to.

Throws:

`java.io.IOException -` for I/O exception

---

encodeBase64Binary

```
public static java.lang.String encodeBase64Binary(byte[] data)
```

Encodes bytes into base64 ASCII characters using the algorithm defined in RFC2045, as defined in w3c stadard http://www.w3.org/tr/2001/rec-xmlschema-2-20010502#base64Binary

Parameters:

`data -` Array containing bytes to be encoded

Returns:

Encoded ASCII string

---

equals

```
public boolean equals(byte[] value)
```

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

Parameters:

`value -` Byte array containing octet data

Returns:

true if equal

---

equals

```
public boolean equals(java.lang.String s)
```

This method compares the given ASN.1 OCTET STRING value to this OCTET STRING value for equality.

Parameters:

`s -` Textual representation of an OCTET STRING. Examples of valid value formats are as follows: Binary string: '11010010111001'B Hex string: '0fa56920014abc'H

Returns:

true if s represents the same OCTET STRING as this OCTET STRING.

---

equals

```
public boolean equals(java.lang.Object os)
```

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

Parameters:

(continued from last page)

`os` - Asn1OctetString object to compare

hashCode

```
public int hashCode()
```

This method returns the hashcode for the Asn1OctetString.

getLength

```
public int getLength()
  throws Asn1InvalidLengthException
```

This method will return the length of the OCTET STRING in octets.

Returns:

Number of octets.

toInputStream

```
public java.io.InputStream toInputStream()
```

This method will return a byte array input stream representation of the octet string value.

Returns:

Reference to input stream object

toString

```
public java.lang.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

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variable within this class.

(continued from last page)

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variable within this class.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1MderOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

Encode this octet string into the MDER encoding. This should be used for octet string types that are not of fixed length.

Throws:

`Asn1MderUnsupported` - if the actual length > 65535 (maximum allowed by MDER).

---

encode

```
public void encode(Asn1MderOutputStream out,
          int constrainedLength)
  throws Asn1Exception,
         java.io.IOException
```

Encode this octet string into the MDER encoding.

Parameters:

`out` - The stream to encode to.
`constrainedLength` - The constrained length of the type being encoded. Pass -1 if the type is unconstrained.

Throws:

`java.io.IOException` - for I/O exception
`Asn1ConsVioException` - if a constrained length is given and the value is not exactly that length.
`Asn1MderUnsupported` - if the length is unconstrained and the actual length > 65535 (maximum allowed by MDER).

com.objsys.asn1j.runtime
Class Asn1OctetStringTagComp

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1OctetStringTagComp
```

All Implemented Interfaces:
java.util.Comparator

---

public class **Asn1OctetStringTagComp**
extends java.lang.Object
implements java.util.Comparator

Comparator for octet strings to compare them as DER encodings, sorting according to the outermost tag.

Constructor Summary

public	Asn1OctetStringTagComp()

Method Summary

int	compare(java.lang.Object a,java.lang.Object b)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

Asn1OctetStringTagComp

```
public Asn1OctetStringTagComp()
```

Methods

compare

```
public int compare(java.lang.Object a,
         java.lang.Object b)
```

com.objsys.asn1j.runtime
Class Asn1OerDecodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1DecodeBitBuffer
               ┗-com.objsys.asn1j.runtime.Asn1OerDecodeBuffer
```

public class **Asn1OerDecodeBuffer**
extends Asn1DecodeBitBuffer

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`
`mTraceHandler`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

`public`	`Asn1OerDecodeBuffer(byte[] msgdata)` This constructor creates a BER decode buffer object that references an encoded ASN.1 message.
`public`	`Asn1OerDecodeBuffer(java.io.InputStream istream)` This constructor creates a BER decode buffer object that references an encoded ASN.1 message.

Method Summary

`int`	`decodeEnumValue()` Decode enumerated value encoded in OER.
`long`	`decodeIntSigned()` Decode an integer value as a variable length, signed integer, including decoding the length, according to OER.

long	`decodeIntSigned(int octets)` Decode a signed integer value (2's complement form), of the given length.
long	`decodeIntUnsigned()` Decode an integer value as a variable length, unsigned integer, including decoding the length, according to OER.
long	`decodeIntUnsigned(int octets)` Decode an unsigned integer value (a binary integer, not 2's complement form), of the given length.
int	`decodeLength()` Decode OER length determinant and return the decoded value.
int	`decodeQuantity()` Decode an OER quantity (used for SEQUENCE-OF and SET-OF)
void	`decodeTag(Asn1Tag tag)` Decode a tag encoded in OER.
boolean	`getCanonicalMode()` Return true if canonical mode has been indicated by calling setCanonicalMode(true);
void	`setCanonicalMode(boolean value)` Turn canonical mode on/off.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`

`binDump, binDump, byteAlign, decodeBit, decodeBitsToInt, decodeBitsToLong, decodeBitsToOctetArray, decodeBitsToOctetArray, getAvailableBits, getBitOffset, getBitOffsetInByte, getMsgBitCnt, getTraceHandler, hasMoreBits, mark, moveBitCursor, readByte, reset, setInputStream, skipBits`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`

`addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBuffer`

`addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`

`getContext, hexDump, hexDump, setKey, setTypeCode`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1OerDecodeBuffer

`public` **`Asn1OerDecodeBuffer`**`(byte[] msgdata)`

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

Parameters:

`msgdata` - Byte array containing an encoded ASN.1 message.

Asn1OerDecodeBuffer

`public` **`Asn1OerDecodeBuffer`**`(java.io.InputStream istream)`

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

Parameters:

`istream` - Input stream containing an encoded ASN.1 message.

Methods

decodeEnumValue

`public int` **`decodeEnumValue`**`()`
`throws java.io.IOException`

Decode enumerated value encoded in OER.

decodeLength

`public int` **`decodeLength`**`()`
`throws java.io.IOException`

Decode OER length determinant and return the decoded value.

decodeIntSigned

`public long` **`decodeIntSigned`**`(int octets)`
`throws java.io.IOException`

Decode a signed integer value (2's complement form), of the given length. Note: this function will do sign-extension so that if a negative value was encoded in less than 8 bytes, the returned long will be that value (and not some positive value).

Parameters:

`octets;` - 0 < octets <= 8

Returns:

decodeIntUnsigned

`public long` **`decodeIntUnsigned`**`(int octets)`
`throws java.io.IOException`

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

Parameters:

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

(continued from last page)

Returns:

decodeIntSigned

```
public long decodeIntSigned()
  throws java.io.IOException
```

Decode an integer value as a variable length, signed integer, including decoding the length, according to OER. This is used for integer values that have a lower bound less than -2^63, no lower bound, or a lower bound less than zero in combination with an upper bound greater than 2^63-1, or no upper bound. (In other words, it doesn't fit in a signed 64-bit integer.)

decodeIntUnsigned

```
public long decodeIntUnsigned()
  throws java.io.IOException
```

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

decodeQuantity

```
public int decodeQuantity()
  throws java.io.IOException
```

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

Returns:

Throws:

`IOException`

decodeTag

```
public void decodeTag(Asn1Tag tag)
  throws java.io.IOException
```

Decode a tag encoded in OER.

Parameters:

`tag` - Object to decode into.

getCanonicalMode

```
public boolean getCanonicalMode()
```

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

Returns:

setCanonicalMode

```
public void setCanonicalMode(boolean value)
```

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

com.objsys.asn1j.runtime
Interface Asn1OerDecoder

public interface **Asn1OerDecoder**

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

Method Summary

`abstract Asn1Type`	`decode(Asn1OerDecodeBuffer buffer)` Decode value from given buffer.

Methods

decode

```
public abstract Asn1Type decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode value from given buffer.

Parameters:

`buffer`

Returns:

com.objsys.asn1j.runtime
Class Asn1OerEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeBitBuffer
               ┗-com.objsys.asn1j.runtime.Asn1OerEncodeBuffer
```

public class **Asn1OerEncodeBuffer**
extends Asn1EncodeBitBuffer

This class handles the encoding of ASN.1 messages as specified in the Octet Encoding Rules (OER) as specified in the ITU-T X.696 standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBitBuffer`
`mByteIndex`, `mData`, `mTraceHandler`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`
`INITIAL_SIZE`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1OerEncodeBuffer()` This constructor creates a OER encode buffer object with the default initial size.
public	`Asn1OerEncodeBuffer(int size)` This constructor creates a OER encode buffer object with the given initial size.

Method Summary

`Asn1OerEncodeBuffer`	`beginSetOf()` If canonical mode is on, return a new Asn1OerEncodeBuffer into which the repetitions of a SET OF can be encoded.
`void`	`binDump(java.io.PrintStream out,java.lang.String varName)` This method dumps the encoded message in a human-readable format to the given print output stream.

void	`encodeIdentifier(int ident)` Encode an identifier according to OER.
void	`encodeIntSigned(long value)` Encode an integer value as a variable length, signed integer, including encoding the length, according to OER.
void	`encodeIntSigned(long value,int octets)` Encode an integer value as a signed value (2's complement form), in the given number of octets.
void	`encodeIntUnsigned(long value)` Encode an integer value as a variable length, unsigned integer, including encoding the length, according to OER.
void	`encodeIntUnsigned(long value,int octets)` Encode an integer value as an unsigned value (binary integer) in the given number of octets.
void	`encodeLength(long length)` Encode an OER length determinant
void	`encodeQuantity(int quantity)` Encode an OER quantity (used for SEQUENCE-OF and SET-OF).
void	`encodeTag(short tagClass,int tagNumber)` Encode a tag according to OER.
Asn1OerEncodeBuffer	`endSetOf()` If canonical mode is on, encode each of the SET OF occurrences, in sorted order, to the original buffer, the one that created this buffer.
boolean	`getCanonicalMode()` Return true if canonical mode has been indicated by calling setCanonicalMode(true);
int	`getIdentifierLength(int ident)` Return the minimal number of octets required to encode the given identifier, where the identifier is encoded following the OER rules for encoding a tag number greater than 62, i.e.
Asn1OerEncodeBuffer	`newBuffer()` Return a new OER encode buffer with the same canonical mode setting as this buffer.
void	`setCanonicalMode(boolean value)` Turn canonical mode on/off.
void	`setOfRepDone()` Invoke this method to signal to the buffer that another occurrence in a SET OF has been completely encoded into it.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBitBuffer`

binDump, byteAlign, checkSize, copy, copy, encodeBit, encodeBits, encodeBits, encodeBits, encodeLongBits, encodeLongBits, getBitOffsetInByte, getBuffer, getByteArrayInputStream, getByteIndex, getMsgBitCnt, getMsgByteCnt, getMsgCopy, getMsgLength, getTraceHandler, hexDump, isByteAligned, reset, reverseBytes, setMsgBitCnt, toString, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1OerEncodeBuffer

`public` **`Asn1OerEncodeBuffer`**`()`

This constructor creates a OER encode buffer object with the default initial size. Whenever the buffer becomes full, it will be expanded.

Asn1OerEncodeBuffer

`public` **`Asn1OerEncodeBuffer`**`(int size)`

This constructor creates a OER encode buffer object with the given initial size. Whenever the buffer becomes full, it will be expanded. For best performance, this size should be large enough to prevent resizing in normal operation.

Parameters:

`size` - The initial size in bytes of an encode buffer.

Methods

binDump

```
public void binDump(java.io.PrintStream out,
                    java.lang.String varName)
```

This method dumps the encoded message in a human-readable format to the given print output stream.

(continued from last page)

beginSetOf

`public Asn1OerEncodeBuffer beginSetOf()`

If canonical mode is on, return a new Asn1OerEncodeBuffer into which the repetitions of a SET OF can be encoded. This should be paired with a later call to endSetOf (which see). If canonical mode is off, this simply returns this buffer. Example usage: buffer = buffer.beginSetOf(); //buffer is now new, temp buffer for each repetition encode repetition to buffer buffer.setOfOccurrenceDone(); buffer = buffer.endSetOf(); //buffer is original buffer again //and SET OF is encoded, sorted, into //it.

Returns:

---

endSetOf

`public Asn1OerEncodeBuffer endSetOf()`

If canonical mode is on, encode each of the SET OF occurrences, in sorted order, to the original buffer, the one that created this buffer. If canonical mode is off, this simply returns itself.

Returns:

In canonical mode, the original buffer, the one that was used to create this buffer by calling beginSetOf(). Otherwise, this.

---

setOfRepDone

`public void setOfRepDone()`

Invoke this method to signal to the buffer that another occurrence in a SET OF has been completely encoded into it. This has no effect if canonical mode is off.

---

encodeIntSigned

```
public final void encodeIntSigned(long value,
                                  int octets)
```

Encode an integer value as a signed value (2's complement form), in the given number of octets. The given value must be able to fit in the given number of octets.

Parameters:

`value` - The value to encode. It must fit in the given number of octets.

`octets` - The number of octets to encode in; 0 < octets <= 8

---

encodeIntUnsigned

```
public final void encodeIntUnsigned(long value,
                                    int octets)
```

Encode an integer value as an unsigned value (binary integer) in the given number of octets.

Parameters:

`value` - The value to encode. It must be non-negative and fit in the given number of octets.

`octets` - The number of octets to encode in; 0 < octets <= 8.

---

encodeIntSigned

`public final void encodeIntSigned(long value)`

Encode an integer value as a variable length, signed integer, including encoding the length, according to OER. This is used for integer values that have a lower bound less than -2^63, no lower bound, or a lower bound less than zero in combination with an upper bound greater than 2^63-1, or no upper bound. (In other words, it doesn't fit in a signed 64-bit integer.)

Parameters:

`value` - The value to encode.

encodeIntUnsigned

`public final void encodeIntUnsigned(long value)`

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

Parameters:

`value` - The value to encode. It must be non-negative.

encodeLength

`public final void encodeLength(long length)`

Encode an OER length determinant

Parameters:

`length`

encodeQuantity

`public final void encodeQuantity(int quantity)`

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

Parameters:

`quantity`

getIdentifierLength

`public final int getIdentifierLength(int ident)`

Return the minimal number of octets required to encode the given identifier, where the identifier is encoded following the OER rules for encoding a tag number greater than 62, i.e. in a variable number of octets, with 7 bits of the value encoded in each octet and the first bit serving as a flag bit.

Parameters:

`ident`

encodeIdentifier

`public final void encodeIdentifier(int ident)`

Encode an identifier according to OER. This is the encoding used for a tag number > 62, i.e. in a variable number of octets, with 7 bits of the value encoded in each octet and the first bit serving as a flag bit. It is also the encoding used for the subidentifiers in an OBJECT IDENTIFIER encoding.

Parameters:

`ident` - The tag number.

encodeTag

```
public final void encodeTag(short tagClass,
                            int tagNumber)
```

Encode a tag according to OER.

Parameters:

`tagClass` - The tag class. The highest 2 bits shall be set equal to the bits specified in the encoding rules for the tag's class. The remaining bits shall be zero. Asn1Tag.UNIV etc. fulfill this.
`tagNumber` - The tag number.

getCanonicalMode

`public boolean` **`getCanonicalMode`**`()`

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

Returns:

newBuffer

`public Asn1OerEncodeBuffer` **`newBuffer`**`()`

Return a new OER encode buffer with the same canonical mode setting as this buffer.

Returns:

setCanonicalMode

`public void` **`setCanonicalMode`**`(boolean value)`

Turn canonical mode on/off. Turning canonical mode on acts as a signal to both generated code and runtime code that the user wants to encode according to the canonical OER rules.

com.objsys.asn1j.runtime
Class Asn1OID_IRI

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1OID_IRI
```

Direct Known Subclasses:
Asn1RELATIVE_OID_IRI

---

public class **Asn1OID_IRI**
extends Asn1CharString

This class represents the ASN.1 OID-IRI. It is really nothing more than a string, but it provides the decoding and encoding methods for various encoding rules. That this class extends Asn1CharString should be considered an implementation detail. The reason for the inheritance is that, like Asn1CharString, this class is a wrapper around a String value.

Field Summary

`static final Asn1Tag`	`TAG` The **TAG**constant describes the universal tag for this data type (UNIVERSAL 35).

Fields inherited from class `com.objsys.asn1j.runtime.Asn1CharString`
`mStringBuffer`, `value`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`BIT_STRING`, `BMPString`, `BOOLEAN`, `DATE`, `DATE_TIME`, `DURATION`, `ENUMERATED`, `EOC`, `EXTERNAL`, `GeneralString`, `GeneralTime`, `GraphicString`, `IA5String`, `INTEGER`, `mNonParameterizedTypeName`, `NULL`, `NumericString`, `OBJECT_IDENTIFIER`, `ObjectDescriptor`, `OCTET_STRING`, `OID_IRI`, `OpenType`, `PrintableString`, `REAL`, `RELATIVE_OID_IRI`, `RelativeOID`, `SEQUENCE`, `SET`, `T61String`, `TeletexString`, `TIME`, `TIME_OF_DAY`, `UniversalString`, `UTCTime`, `UTF8String`, `VideotexString`, `VisibleString`

Constructor Summary

protected	`Asn1OID_IRI(short typeCode)` The default constructor creates an empty string object.
public	`Asn1OID_IRI()` The default constructor creates an empty string object.
protected	`Asn1OID_IRI(java.lang.String data,short typeCode)` This version of the constructor can be used to set the string **value**member variable to the given string value.

public	`Asn1OID_IRI(java.lang.String data)` This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 OID-IRI string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 OID-IRI string value that was encoded according to the Octet Encodnig Rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 OID-IRI using the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This method decodes an ASN.1 OID-IRI using the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 OID-IRI string value.
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes and writes to the stream an ASN.1 OID-IRI value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 OID-IRI string value according to the Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 OID-IRI value using the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This method encodes an ASN.1 OID-IRI value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 OID-IRI string value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,long lower,long upper)` This method encodes an ASN.1 OID-IRI string value using the packed encoding rules (PER).
Asn1Tag	`getTag()` Allows subclass to use a different tag by overriding this method.
short	`getTagNumber()` Allows subclass to use a different tag number by overriding this method.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 35).

Constructors

Asn1OID_IRI

```
protected Asn1OID_IRI(short typeCode)
```

The default constructor creates an empty string object.

Asn1OID_IRI

```
public Asn1OID_IRI()
```

The default constructor creates an empty string object.

Asn1OID_IRI

```
protected Asn1OID_IRI(java.lang.String data,
                      short typeCode)
```

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

Asn1OID_IRI

```
public Asn1OID_IRI(java.lang.String data)
```

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

Methods

(continued from last page)

getTagNumber

```
protected short getTagNumber()
```

Allows subclass to use a different tag number by overriding this method.

Returns:

---

getTag

```
protected Asn1Tag getTag()
```

Allows subclass to use a different tag by overriding this method.

Returns:

---

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 OID-IRI 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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 OID-IRI using the packed encoding rules (PER).

Parameters:

`buffer` - Decode message buffer object

(continued from last page)

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

This method decodes an ASN.1 OID-IRI using the packed encoding rules (PER).

Parameters:

`buffer` - Decode message buffer object
`lower` - ignored
`upper` - ignored

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an ASN.1 OID-IRI value using the packed encoding rules (PER). The value to be encoded is stored in the 'value' public member variable within this class.

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an ASN.1 OID-IRI value using the packed encoding rules (PER). The value to be encoded is stored in the 'value' public member variable within this class.

Parameters:

`buffer` - Encode message buffer object
`lower` - ignored
`upper` - ignored

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 OID-IRI string value that was encoded according to the Octet Encodnig Rules (OER).

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 OID-IRI string value according to the Octet Encoding Rules (OER).

Parameters:

`buffer` - Encode message buffer object

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 OID-IRI string value using the packed encoding rules (PER). The value to be encoded is stored in the 'value' public member variable within this class.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 OID-IRI string value using the packed encoding rules (PER). The value to be encoded is stored in the 'value' public member variable within this class.

Parameters:

`out` - PER Output Stream object

`lower` - ignored

`upper` - ignored

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1OpenExt

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OpenExt
```

public class **Asn1OpenExt**
extends Asn1Type

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).

Field Summary

java.util.ArrayList	value The value is a list of Asn1OpenType objects.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1OpenExt()

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).
void	decode(Asn1PerDecodeBuffer buffer) This method decodes an open type extension in a SEQUENCE or SET construct using the packed encoding rules (PER).
void	decodeComponent(Asn1BerDecodeBuffer buffer) 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.
void	decodeEventComponent(Asn1BerDecodeBuffer buffer) 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 "..."

void	`decodeExtension(Asn1JsonDecodeBuffer buffer,java.lang.String name)` Decode a single occurrence of an extension from JSON and add an Asn1OpenType for it to the list of extensions.
Asn1OpenType	`decodeOpenType(Asn1OerDecodeBuffer buffer,boolean present,int index)` This method decodes a single open type extension item in a SEQUENCE or SET construct using the octet encoding rules (OER).
Asn1OpenType	`decodeOpenType(Asn1PerDecodeBuffer buffer,boolean present,int index)` This method decodes a single open type extension item in a SEQUENCE or SET construct using the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER).
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER) and writes it into the stream.
void	`encode(Asn1JsonOutputStream outstream,boolean asArray)` Encode the open extension elements to JSON
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes the ASN.1 open type extensions using Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 open type extension value using the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncoder buffer)` This method encodes an ASN.1 open type extension value using the XML Encoding Rules (XER).
void	`encode(Asn1XmlEncoder buffer)` This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard (asn2xsd).
void	`encodeExtBits(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 open type extension value bits using the Packed Encoding Rules (PER).
void	`encodeExtBits(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 open type extension value bits using the Packed Encoding Rules (PER).
boolean	`equals(java.lang.Object o)` Compares extension lists.
java.lang.String	`getAsn1TypeName()` This method returns "Open Type Extension" as the ASN.1 type name.

int	hashCode() Returns the hashcode for an extension list.
boolean	hasPresentExtensions() Return true if this contains present extensions.
void	setOpenType(Asn1OpenType object,int index) This method will add the given open type object to the open extension element list at the given index.
java.lang.String	toString() This method will return a string representation of the open extension value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

value

public transient java.util.ArrayList **value**

The value is a list of Asn1OpenType objects. Each of these objects contains a fully encoded extension item.

Constructors

Asn1OpenExt

public **Asn1OpenExt**()

Methods

getAsn1TypeName

public java.lang.String **getAsn1TypeName**()

This method returns "Open Type Extension" as the ASN.1 type name. It is overridden because open type extensions cannot set their type names in generated code.

hasPresentExtensions

public boolean **hasPresentExtensions**()

(continued from last page)

Return true if this contains present extensions. Null values in the value list represent absent extensions.

Returns:

---

encodeExtBits

```
public void encodeExtBits(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

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

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes the ASN.1 open type extensions using Octet Encoding Rules (OER).

Parameters:

`buffer` - Encode message buffer object

---

decodeEventComponent

```
public void decodeEventComponent(Asn1BerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

decodeComponent

```
public void decodeComponent(Asn1BerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
          boolean explicit,
          int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - Decode message buffer object
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length if implicit element

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object
`explicit` - Flag indicating element is explicitly tagged

Returns:

Length of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`public member list variable. If optional items are absent, null placeholders will be inserted in the list.

Parameters:

`buffer` - Decode message buffer object

decodeOpenType

```
public Asn1OpenType decodeOpenType(Asn1PerDecodeBuffer buffer,
         boolean present,
         int index)
  throws Asn1Exception,
         java.io.IOException
```

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

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`buffer` - Encode message buffer object

encodeExtBits

```
public void encodeExtBits(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

(continued from last page)

Parameters:

`buffer` - Encode message buffer object

---

decodeOpenType

```
public Asn1OpenType decodeOpenType(Asn1OerDecodeBuffer buffer,
          boolean present,
          int index)
  throws Asn1Exception,
          java.io.IOException
```

This method decodes a single open type extension item in a SEQUENCE or SET construct using the octet encoding rules (OER). 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. If index >= value.size, the item is appended to the list (all such values of index thus have the same behavior).

---

setOpenType

```
public void setOpenType(Asn1OpenType object,
          int index)
```

This method will add the given open type object to the open extension element list at the given index.

Parameters:

`object` - Open type object

`index` - Index in open type list where element is to be placed

---

toString

```
public java.lang.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

---

encode

```
public void encode(Asn1BerOutputStream out,
          boolean explicit)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an ASN.1 open type extension value using the Basic Encoding Rules (BER) and writes it into the stream.

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating element is explicitly tagged

Throws:

`IOException` - Any exception thrown by the Asn1BerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1XerEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1XmlEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

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

---

decodeExtension

```
public void decodeExtension(Asn1JsonDecodeBuffer buffer,
         java.lang.String name)
  throws java.io.IOException
```

Decode a single occurrence of an extension from JSON and add an Asn1OpenType for it to the list of extensions.

Parameters:

`buffer` - Decode message buffer object
`name` - The name of the extension element (previously decoded). This becomes a part of the character data held in the Asn1OpenType object, so that the character data represents a full JSONNamedValue. Pass null for extensions in [ARRAY] SEQUENCE.

---

encode

```
public void encode(Asn1JsonOutputStream outstream,
         boolean asArray)
  throws java.io.IOException
```

Encode the open extension elements to JSON

Parameters:

`asArray` - true if the enclosing type is [ARRAY] SEQUENCE; false otherwise.

(continued from last page)

equals

`public boolean equals(java.lang.Object o)`

Compares extension lists. Returns true under the following conditions: The extension list is compared to itself.The contents of the extensions are identical.

Parameters:

`o` - The object to compare.

Returns:

True on success, false on failure.

---

hashCode

`public int hashCode()`

Returns the hashcode for an extension list.

com.objsys.asn1j.runtime
Class Asn1OpenType

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
         ┗-com.objsys.asn1j.runtime.Asn1OpenType
```

Direct Known Subclasses:
Asn1XerOpenType, Asn1ChoiceExt

---

public class **Asn1OpenType**
extends Asn1OctetString

This is a container class for holding an ASN.1 open type value.

Where the data is internally stored and how it is interpreted is controlled by a "dataEncoding" field. You can specify the data encoding when creating the object. It will also be set when decoding. The following explains the possible data encodings:
- UNKNOWN: `value` is used to hold byte data. It is interpreted as the encoding of the actual value according to some unknown encoding rules (BER, PER, XER, JSON, or some other).
- BER, PER, OER: `value` is used to hold byte data. It is interpreted as the encoding of the actual value according to BER, PER, or OER, respectively.
- XER: `value` is used to hold byte data. It is interpreted as the the encoding of the actual value in XML (whether X.693 or Obj-Sys rules), where the XML characters are encoded to bytes using the UTF-8 character encoding.
- JSON: a private field is used to hold character data. The character data is the encoding of the actual value in JSON.

Note especially that the data encoding indicates the encoding rules used to encode the actual value. That result is typically encoded as part of some larger encoding. The data encoding does NOT indicate the rules used for that larger encoding. For example, using an xmlhstring representation, XER and JSON will encode an open type that was previously encoded using any arbitrary encoding rules. When decoding such data, the data encoding will be UNKNOWN, not XER nor JSON.

Field Summary

static final int	BER Value: **1**
int	dataEncoding Specifies the nature of the data held by this object.
static final java.lang.String	EDATAMSG Value: **ENCODED DATA**
static final int	JSON Value: **4**
Asn1EncodeBuffer	mEncodeBuffer
int	mLength

static final int	OER Value: **5**
static final int	PER Value: **2**
static final int	UNKNOWN Value: **0**
static final int	XER Value: **3**

Fields inherited from class com.objsys.asn1j.runtime.Asn1OctetString
TAG, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1OpenType() This constructor creates an empty type that can be used in a decode method call to receive an encoded value.
public	Asn1OpenType(byte[] data) This constructor initializes an open type from the given byte array.
public	Asn1OpenType(byte[] data,int encoding) This constructor initializes an open type from the given byte array.
public	Asn1OpenType(byte[] data,int offset,int nbytes) This constructor initializes the open type from a portion of the given byte array.
public	Asn1OpenType(byte[] data,int offset,int nbytes,int encoding) This constructor initializes the open type from a portion of the given byte array.
public	Asn1OpenType(Asn1EncodeBuffer buffer) This constructor initializes an open type using an encoded component.
public	Asn1OpenType(int length) **Deprecated.**
public	Asn1OpenType(java.lang.String data,int encoding) Convenience constructor.

public	`Asn1OpenType(char[] data,int encoding)` Create Asn1OpenType on the given character data.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 open type value.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 open type value from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 open type value using the Octet Encoding Rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 open type value using the packed encoding rules (PER).
void	`decode72(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 open type value from JSON following rules created by Obj-Sys before X.697.
void	`decodeExtension(Asn1JsonDecodeBuffer buffer,java.lang.String name)` Decode an extension from JSON.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 open type value.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)`
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 open type value using the Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncoder buffer)` This method encodes an ASN.1 open type value using the XML Encoding Rules (XER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 open type value using the XML Encoding Rules (XER).
void	`encode(Asn1XmlEncoder buffer)` This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd).

void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	encode72(Asn1JsonOutputStream outstream) Encode to JSON folllowing rules created by Obj-Sys before X.697.
void	encodeAsExtension(Asn1JsonOutputStream outstream) This method encodes an extension element to JSON.
void	encodeAsExtension(Asn1XerEncoder buffer) This method encodes an extension element to XML.
void	encodeAsExtension(Asn1XmlEncoder buffer) This method encodes an extension element to XML.
java.lang.String	getAsn1TypeName() This method returns "Opaque Open Type" as the ASN.1 type name.
char[]	getCharData() Returns the character data for the open type.
int	getDataEncoding() Return the data encoding of the data.
Asn1XerSaxHandler	getSaxHandler() Get a SAX handler for this object.
Asn1XerSaxHandler	getSaxHandler(boolean captureOuterElem) Get a SAX handler for this object.
void	setBinaryData(byte[] data,int encoding) Sets the binary data for the open type and assigns the data encoding to the given encoding.
void	setCharData(char[] data,int encoding) Sets the character data for the open type and assigns the data encoding to the given encoding.
void	setCharData(java.lang.String data,int encoding) Convenience method; same as setCharData(value.toCharArray(), encoding)
java.lang.String	toString() This method will return a string representation of the open type value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1OctetString

compareTo, decode, decode, decode, decode, decode, decode, decode, decode, decodeAsBase64, decodeAsHex, decodeContent, decodeRemainingBits, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsBase64, encodeAsHex, encodeAttribute, encodeBase64Binary, encodeContent, equals, equals, equals, getAsn1TypeName, getLength, getMderLength, hashCode, toInputStream, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

UNKNOWN

public static final int **UNKNOWN**

BER

public static final int **BER**

PER

public static final int **PER**

XER

public static final int **XER**

JSON

public static final int **JSON**

OER

public static final int **OER**

mLength

protected transient int **mLength**

(continued from last page)

mEncodeBuffer

```
protected transient com.objsys.asn1j.runtime.Asn1EncodeBuffer mEncodeBuffer
```

dataEncoding

```
protected int dataEncoding
```

Specifies the nature of the data held by this object.

EDATAMSG

```
protected static final java.lang.String EDATAMSG
```

Constructors

Asn1OpenType

```
public Asn1OpenType()
```

This constructor creates an empty type that can be used in a decode method call to receive an encoded value. The data encoding is UNKNOWN.

Asn1OpenType

```
public 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. The data encoding is UNKNOWN.

Parameters:

`data` - Byte array containing a previously encoded value.

Asn1OpenType

```
public Asn1OpenType(byte[] data,
                    int encoding)
```

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 value.
`encoding` - The encoding that describes the meaning of data. Any of the encodings other than JSON.

Asn1OpenType

```
public 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. The data encoding is UNKNOWN.

Parameters:

`data` - Byte array containing a previously encoded value.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array

Asn1OpenType

```
public Asn1OpenType(byte[] data,
                    int offset,
                    int nbytes,
                    int encoding)
```

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 a previously encoded value.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array
`encoding` - The encoding that describes the meaning of data. Any of the encodings other than JSON.

Asn1OpenType

```
public 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. The data encoding is derived from the type of Asn1EncodeBuffer given, and is possibly UNKNOWN.

Parameters:

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

Asn1OpenType

```
public Asn1OpenType(int length)
```

Deprecated.

This constructor initializes an open type using just an encoded component length. This can be used if a header (for example, a ROSE header) is being prepended to a pre-encoded component. This form of the constructor is only being maintained for compatibility with previous versions. The preferred way to use a pre-encoded component is to use the version of the constructor that takes an Asn1BerEncodeBuffer argument. The result is the same, but the print method will not work if this version is used. The data encoding is UNKNOWN.

Parameters:

`length` - Length of the pre-encoded component. This must be the last item in the type and already exist in the encode buffer.

Asn1OpenType

```
public Asn1OpenType(java.lang.String data,
                    int encoding)
```

Convenience constructor. Same as Asn1OpenType(data.toCharArray(), encoding)

Asn1OpenType

```
public Asn1OpenType(char[] data,
                    int encoding)
```

Create Asn1OpenType on the given character data.

Parameters:

`data` - The character data array.
`encoding` - The encoding. Either XER or JSON. If JSON, data is taken to be the JSON encoding of the actual value. The array is not copied, so beware using it after passing it here. The `value` field will be assigned null. If XER, data is taken to be the XER encoding of the actual value. The `value` field will be assigned the UTF-8 character encoding of the given characters.

Methods

getAsn1TypeName

`public java.lang.String **getAsn1TypeName**()`

This method returns "Opaque Open Type" as the ASN.1 type name.

---

getCharData

`public char[] **getCharData**()`

Returns the character data for the open type. Currently, this is only supported when getDataEncoding() returns JSON.

Returns:

The char data. The actual internal array is returned; a copy is not made.

Throws:

`RuntimeException` - if data encoding indicates there is no char data.

---

setBinaryData

```
public void setBinaryData(byte[] data,
          int encoding)
```

Sets the binary data for the open type and assigns the data encoding to the given encoding.

Parameters:

`data` - The binary data that make up an encoding of the actual value.

`encoding` - Identifies the encoding rules for which data is an encoding. This can be any of the encodings except JSON.

---

setCharData

```
public void setCharData(java.lang.String data,
          int encoding)
```

Convenience method; same as setCharData(value.toCharArray(), encoding)

---

setCharData

```
public void setCharData(char[] data,
          int encoding)
```

Sets the character data for the open type and assigns the data encoding to the given encoding.

Parameters:

`data` - The character data that make up an encoding of the actual value.

`encoding` - Permissible values are JSON and XER. If XER, the given data will be converted to bytes using UTF-8 and held in the `value` field. If JSON, the given data will be held as-is. The array will not be copied. The `value` field will be assigned null.

---

getDataEncoding

`public int **getDataEncoding**()`

Return the data encoding of the data.

Returns:

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 open type value. The data encoding will be set to BER.

Parameters:

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

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

This method encodes an ASN.1 open type value. The value is assumed to be an already-encoded BER message component and will be copied to the encoded buffer. An optimization is available in which no copy will performed if the encoded component is already present in the encode buffer. This is done if the form of constructor specifiying only a component length is used.

Parameters:

`buffer` - Encode message buffer object
`explicit` - Flag indicating element is explicitly tagged

Returns:

Length of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 open type value using the packed encoding rules (PER). The data encoding will be set to PER.

Parameters:

`buffer` - Decode message buffer object

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER). The data should be the value pre-encoded encoded in PER.

Parameters:

`buffer` - Encode message buffer object

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 open type value using the Octet Encoding Rules (OER). This object will subsequently contain the encoding of the open type (which should be OER).

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 open type value using the Octet Encoding Rules (OER). The data should be the value pre-encoded in OER.

Parameters:

`buffer` - Encode message buffer object

encodeAsExtension

```
public void encodeAsExtension(Asn1XmlEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an extension element to XML. If the data encoding is other than XER, the hexadecimal representation of the data is encoded inside an XML comment.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd). If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters:

`buffer` - Encode message buffer object
`elemName` - Ignored
`nsPrefix` - Ignored

encode

```
public void encode(Asn1XmlEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 open type value using the XML Encoding as specified in the XML schema standard(asn2xsd). If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 open type value using the XML Encoding Rules (XER). If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

(continued from last page)

Parameters:

`buffer` - Encode message buffer object

`elemName` - Ignored

encodeAsExtension

```
public void encodeAsExtension(Asn1XerEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an extension element to XML. If the data encoding is other than XER, its hexadecimal representation is encoded inside an XML comment.

encode

```
public void encode(Asn1XerEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 open type value using the XML Encoding Rules (XER). If the data encoding is XER, the data is written as-is. Otherwise, the data's hexadecimal representation is encoded (i.e. xmlhstring).

Parameters:

`buffer` - Encode message buffer object

decodeExtension

```
public void decodeExtension(Asn1JsonDecodeBuffer buffer,
          java.lang.String name)
  throws java.io.IOException
```

Decode an extension from JSON. There should be, possibly after some leading whitespace, a JSON value on the input. After decoding, this object's data encoding will be JSON. The character data will be either a JSONValue or JSONNamedValue. If name is null, then it will be a JSONValue; otherwise, it will be a JSONNamedValue, formed from the given (previously decoded) name and the decoded value.

encodeAsExtension

```
public void encodeAsExtension(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

This method encodes an extension element to JSON. If the data encoding is other than JSON, it cannot be encoded and the data is dropped.

decode72

```
public void decode72(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 open type value from JSON following rules created by Obj-Sys before X.697. If the JSON value was a JSONNestedValue, the data encoding will be JSON, and the data will be character data. If the JSON value was a JSON string (a quoted xmlhstring), the data encoding will be UNKNOWN and the data will be byte data. For decoding an unknown extension element, use decodeExtension.

Parameters:

`buffer`

Throws:

`java.io.IOException`

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

(continued from last page)

Decode ASN.1 open type value from JSON. For decoding an unknown extension element, use decodeExtension.

Parameters:

`buffer`

Throws:

`java.io.IOException`

---

encode72

```
public void encode72(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode to JSON folllowing rules created by Obj-Sys before X.697.

Parameters:

`outstream`

Throws:

`java.io.IOException`

---

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

---

toString

```
public java.lang.String toString()
```

This method will return a string representation of the open type value. If the data encoding is XER or JSON, the string will consist of the corresponding characters (in the case of XER, the byte data is converted to characters using UTF-8). In all other cases, the hexadecimal representation of the byte data is returned.

Returns:

Stringified representation of the value

---

encode

```
public void encode(Asn1BerOutputStream out,
                   boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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). The data should be the value pre-encoded in BER.

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 open type value using the Packed Encoding Rules (PER). The data should be the value pre-encoded in PER.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

getSaxHandler

```
public Asn1XerSaxHandler getSaxHandler()
```

Get a SAX handler for this object. If this is the first invocation of any of the getSaxHandler methods on this object, the returned handler will capture the outer element start/end tags. Otherwise, this will simply return the same SAX handler as previously returned.

Returns:

---

getSaxHandler

```
public Asn1XerSaxHandler getSaxHandler(boolean captureOuterElem)
```

Get a SAX handler for this object.

Parameters:

`captureOuterElem` - Pass true if the outer element start and end tags (if present) should be captured. The outer element is the element for which startElement is called when the level is the start level. Note that this parameter is ignored if you have previously invoked one of the getSaxHandler methods on this object.

Returns:

com.objsys.asn1j.runtime
Class Asn1OpenType.SaxHandler

```
java.lang.Object
   ┗-org.xml.sax.helpers.DefaultHandler
      ┗-com.objsys.asn1j.runtime.Asn1XerSaxHandler
         ┗-com.objsys.asn1j.runtime.Asn1OpenType.SaxHandler
```

public class **Asn1OpenType.SaxHandler**
extends Asn1XerSaxHandler

Fields inherited from class com.objsys.asn1j.runtime.Asn1XerSaxHandler
mConsumedStartElement, mCurrElemID, mCurrState, mLevel, mStartLevel, mXMLElemName, XERDATA, XEREND, XERINIT, XERSTART, XERUNKNOWN

Method Summary

void	characters(char[] ch,int start,int length)
void	endElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName)
void	startElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName,org.xml.sax.Attributes atts)

Methods inherited from class com.objsys.asn1j.runtime.Asn1XerSaxHandler
consumeStartElement, endGroup, error, fatalError, getState, init, isComplete, isDecodingAsGroup, matchXMLElemName, setComplete, setLevel, setXMLElemName, warning

Methods inherited from class org.xml.sax.helpers.DefaultHandler
characters, endDocument, endElement, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator, skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

(continued from last page)

startElement

```
public void startElement(java.lang.String namespaceURI,
          java.lang.String localName,
          java.lang.String qName,
          org.xml.sax.Attributes atts)
  throws org.xml.sax.SAXException
```

---

characters

```
public void characters(char[] ch,
          int start,
          int length)
  throws org.xml.sax.SAXException
```

---

endElement

```
public void endElement(java.lang.String namespaceURI,
          java.lang.String localName,
          java.lang.String qName)
  throws org.xml.sax.SAXException
```

com.objsys.asn1j.runtime
Class Asn1OpenTypeField

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1OpenTypeField
```

public class **Asn1OpenTypeField**
extends java.lang.Object

Asn1OpenTypeField carries information about the actual type for an open type field in an information object.

Field Summary

`java.lang.Class`	`actualType` actualType is the Class for the actual type.
`java.lang.Object`	`decoder` decoder is some object which may implement known interfaces for decoding an object from an encoding (e.g.
`java.lang.String`	`nonParameterizedTypeName` nonParameterizedTypeName is the NonParameterizedTypeName for the actual type, as specified in X.681.

Constructor Summary

`public`	`Asn1OpenTypeField(java.lang.Class actualType,java.lang.Object decoder,java.lang.String nonParameterizedTypeName)`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

actualType

`public java.lang.Class` **`actualType`**

actualType is the Class for the actual type. It can be used to instantiate the correct Java class so that known decode methods can be invoked. In some cases (e.g. enumerated types), decode methods are not to be invoked, which is why decoder is also provided.

decoder

`public java.lang.Object` **`decoder`**

(continued from last page)

decoder is some object which may implement known interfaces for decoding an object from an encoding (e.g. Asn1BerDecoder, Asn1PerDecoder, Asn1XerDecoder). When not-null and implementing an interface appropriate to the encoding rules in effect, the decoder should be used rather than using actualType to instantiate an object and invoking some decode method on it.

nonParameterizedTypeName

```
public java.lang.String nonParameterizedTypeName
```

nonParameterizedTypeName is the NonParameterizedTypeName for the actual type, as specified in X.681. This is used by an XER encoding, and can be null when no XER encoders are generated.

Constructors

Asn1OpenTypeField

```
public Asn1OpenTypeField(java.lang.Class actualType,
                         java.lang.Object decoder,
                         java.lang.String nonParameterizedTypeName)
```

com.objsys.asn1j.runtime
Class Asn1OutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-com.objsys.asn1j.runtime.Asn1OutputStream
```

Direct Known Subclasses:
Asn1XmlOutputStream, Asn1XerOutputStream, Asn1PerOutputStream, Asn1MderOutputStream, Asn1BerOutputStream

---

public abstract class **Asn1OutputStream**
extends java.io.OutputStream

This abstract class implements the base output stream to encode ASN.1 messages.

Field Summary

java.io.OutputStream	os

Constructor Summary

public	Asn1OutputStream(java.io.OutputStream os) This constructor creates an output stream object.

Method Summary

void	close() Closes this output stream and releases any system resources associated with this stream.
void	flush() Flushes this output stream and forces any buffered output bytes to be written out.
Asn1Context	getContext() Return the context object associated with this stream.
void	write(byte[] b) Writes b.lengthbytes from the specified byte array to this output stream.
void	write(byte[] b,int off,int len) Writes lenbytes from the specified byte array starting at offset offto this output stream.
void	write(int b) Writes the specified byte to this output stream.
void	write2Bytes(int value) Write the lowest two bytes of value to the output stream.

void	write4Bytes(int value) Write the four bytes of value to the output stream.

Methods inherited from class java.io.OutputStream
close, flush, write, write, write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

os

protected java.io.OutputStream **os**

Constructors

Asn1OutputStream

public **Asn1OutputStream**(java.io.OutputStream os)

This constructor creates an output stream object.

Parameters:

os - The underlying OutputStream object.

Methods

getContext

public Asn1Context **getContext**()

Return the context object associated with this stream.

Returns:

The context.

write2Bytes

public void **write2Bytes**(int value)
  throws java.io.IOException

Write the lowest two bytes of value to the output stream. The lowest byte is written last.

Parameters:

value - The value to write.

Throws:

IOException - for any I/O error

write4Bytes

```
public void write4Bytes(int value)
  throws java.io.IOException
```

Write the four bytes of value to the output stream. The lowest byte is written last.

Parameters:

`value` - The value to write.

Throws:

`IOException` - for any I/O error

write

```
public void write(byte[] b)
  throws java.io.IOException
```

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.

Throws:

`IOException` - if an I/O error occurs.

write

```
public void write(byte[] b,
          int off,
          int len)
  throws java.io.IOException
```

Writes `len`bytes from the specified byte array starting at offset `off`to this output stream.

Parameters:

`b` - the data.

`off` - the start offset in the data.

`len` - the number of bytes to write.

Throws:

`IOException` - if an I/O error occurs. In particular, an `IOException`is thrown if the output stream is closed.

write

```
public void write(int b)
  throws java.io.IOException
```

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`.

Throws:

`IOException` - if an I/O error occurs. In particular, an `IOException`may be thrown if the output stream has been closed.

(continued from last page)

close

```
public void close()
  throws java.io.IOException
```

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.

Throws:

`IOException` - if an I/O error occurs.

flush

```
public void flush()
  throws java.io.IOException
```

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.

Throws:

`IOException` - if an I/O error occurs.

com.objsys.asn1j.runtime
Class Asn1PerBitField

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerBitField
```

public class **Asn1PerBitField**
extends java.lang.Object

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

Constructor Summary

public	`Asn1PerBitField(java.lang.String name,int bitOffset,int bitCount)` This constructor initializes all of the variables used to track the bit fields.

Method Summary

int	`getBitCount()` This method returns the number of bits in the bit field.
int	`getBitOffset()` This method returns the offset to the bit field in the buffer.
java.lang.String	`getName()` This method returns the name assigned to the bit field.
void	`setBitCount(int value)` This method sets the count of bits in the bit field.
void	`setBitCountAndOffset(int count,int offset)` This method sets the count of bits in the bit field and the offset to the bit field in the message buffer.
void	`setBitOffset(int value)` This method sets the offset to the bit field in the message buffer.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1PerBitField

```
public Asn1PerBitField(java.lang.String name,
                       int bitOffset,
                       int bitCount)
```

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

Parameters:

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

Methods

getBitCount

```
public int getBitCount()
```

This method returns the number of bits in the bit field.

---

getBitOffset

```
public int getBitOffset()
```

This method returns the offset to the bit field in the buffer.

---

getName

```
public java.lang.String getName()
```

This method returns the name assigned to the bit field.

---

setBitCount

```
public void setBitCount(int value)
```

This method sets the count of bits in the bit field.

---

setBitCountAndOffset

```
public void setBitCountAndOffset(int count,
         int offset)
```

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

---

setBitOffset

```
public void setBitOffset(int value)
```

This method sets the offset to the bit field in the message buffer.

com.objsys.asn1j.runtime
Class Asn1PerBitFieldList

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerBitFieldList
```

public class **Asn1PerBitFieldList**
extends java.lang.Object

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

Constructor Summary

public	`Asn1PerBitFieldList()`

Method Summary

void	`addElemName(java.lang.String name,int arrayx)` This method adds an element name to the current fully qualified name.
Asn1PerBitField	`getCurrBitField()` This method returns a reference to the current bit field object (i.e.
java.util.Iterator	`iterator()` This method returns an iterator to the encapsulated bit field linked list object.
Asn1PerBitField	`newBitField(java.lang.String nameSuffix,int bitOffset,int bitCount)` This method creates a new bit field object with the given properties and appends it to the bit field list.
void	`removeLastElemName()` This method removes the last element name in the current fully qualified name string.
void	`reset()` This method resets the object.
void	`setBitOffset(int bitOffset)` This method is used to change the bit offset in the current bit field.
void	`updateBitFieldLen(int nextBitOffset)` This method updates the bit count of the most recently added bit field, based on the bit offset of the next field.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1PerBitFieldList

```
public Asn1PerBitFieldList()
```

Methods

addElemName

```
public void addElemName(java.lang.String name,
                        int arrayx)
```

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

Parameters:

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

getCurrBitField

```
public Asn1PerBitField getCurrBitField()
```

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

iterator

```
public java.util.Iterator iterator()
```

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

newBitField

```
public Asn1PerBitField newBitField(java.lang.String nameSuffix,
                                   int bitOffset,
                                   int bitCount)
```

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

Parameters:

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

updateBitFieldLen

```
public final void updateBitFieldLen(int nextBitOffset)
```

This method updates the bit count of the most recently added bit field, based on the bit offset of the next field.

Parameters:

`nextBitOffset` - Offset to the start of the next field in bits from the beginning of the encode buffer.

(continued from last page)

removeLastElemName

`public void removeLastElemName()`

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

reset

`public void reset()`

This method resets the object.

setBitOffset

`public void setBitOffset(int bitOffset)`

This method is used to change the bit offset in the current bit field.

com.objsys.asn1j.runtime
Class Asn1PerBitFieldPrinter

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerBitFieldPrinter
```

public class **Asn1PerBitFieldPrinter**
extends java.lang.Object

This class is used to obtain a formatted printout of the bit fields that make up a message. (This was originally for PER but is now used more generally.)

Field Summary

int	mBitMask
int	mByteIndex
int	mCurrOctet
java.io.InputStream	mEncodedMessage
int	mFmtAscCharIdx
int	mFmtBitCharIdx
int	mFmtHexCharIdx
java.lang.StringBuffer	mFormatBuffer
Asn1BitMessageBuffer	mPerMessageBuffer

Constructor Summary

public	Asn1PerBitFieldPrinter(Asn1BitMessageBuffer bitMessageBuffer,java.io.InputStream encodedMessage) Constructor

Method Summary

void	print(java.io.PrintStream out,java.lang.String varName) This method iterates through and prints all of the bit fields in a PER encoded message.

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mBitMask

protected int **mBitMask**

mByteIndex

protected int **mByteIndex**

mCurrOctet

protected int **mCurrOctet**

mFormatBuffer

protected java.lang.StringBuffer **mFormatBuffer**

mFmtBitCharIdx

protected int **mFmtBitCharIdx**

mFmtHexCharIdx

protected int **mFmtHexCharIdx**

mFmtAscCharIdx

protected int **mFmtAscCharIdx**

mEncodedMessage

protected java.io.InputStream **mEncodedMessage**

mPerMessageBuffer

protected com.objsys.asn1j.runtime.Asn1BitMessageBuffer **mPerMessageBuffer**

(continued from last page)

Constructors

Asn1PerBitFieldPrinter

```
public Asn1PerBitFieldPrinter(Asn1BitMessageBuffer bitMessageBuffer,
                              java.io.InputStream encodedMessage)
```

Constructor

Parameters:

`bitMessageBuffer` - PER encode or decode message buffer

`encodedMessage` - Input stream

Methods

print

```
public void print(java.io.PrintStream out,
         java.lang.String varName)
  throws java.io.IOException
```

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

Parameters:

`out` - Print stream

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

com.objsys.asn1j.runtime
Class Asn1PerDecodeBuffer

```
java.lang.Object
   |-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      |-com.objsys.asn1j.runtime.Asn1MessageBuffer
         |-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            |-com.objsys.asn1j.runtime.Asn1DecodeBitBuffer
               |-com.objsys.asn1j.runtime.Asn1PerDecodeBuffer
```

All Implemented Interfaces:
Asn1PerMessageBuffer, Asn1BitMessageBuffer

Direct Known Subclasses:
Asn1PerInputStream

---

public class **Asn1PerDecodeBuffer**
extends Asn1DecodeBitBuffer
implements Asn1BitMessageBuffer, Asn1PerMessageBuffer

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`
`mTraceHandler`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

`public`	`Asn1PerDecodeBuffer(byte[] msgdata,boolean aligned)` This constructor creates a PER decode buffer object that references an encoded ASN.1 message.
`public`	`Asn1PerDecodeBuffer(java.io.InputStream istream,boolean aligned)` This constructor creates a PER decode buffer object that references an encoded ASN.1 message.

Method Summary

void	`byteAlign()` This methods byte-aligns the buffer.
boolean	`decodeBit()` This method decodes a single bit value.
boolean	`decodeBit(java.lang.String ident)` This method decodes a single bit value.
int	`decodeBitsToInt(int nbits)` This method decodes bits from the input stream into a standard integer value.
int	`decodeBitsToInt(int nbits,java.lang.String ident)` This method decodes bits from the input stream into a standard integer value.
long	`decodeBitsToLong(int nbits)` This method decodes bits from the input stream into a long integer value.
long	`decodeBitsToLong(int nbits,java.lang.String ident)` This method decodes bits from the input stream into a long integer value.
void	`decodeBitsToOctetArray(byte[] value,int offset,int nbits)` This method decodes bits from the input stream into an array of octets.
void	`decodeBitsToOctetArray(byte[] value,int offset,int bitOffset,int nbits,java.lang.String ident)` This method decodes bits from the input stream into an array of octets.
void	`decodeBitsToOctetArray(byte[] value,int offset,int nbits,java.lang.String ident)` This method decodes bits from the input stream into an array of octets.
void	`decodeCharString(int nchars,int abpc,int ubpc,Asn1CharSet charSet,java.lang.StringBuffer sbuf)` This method decodes the contents of a known-multiplier character string.
long	`decodeConsWholeNumber(long rangeValue)` This method implements the rules to decode a constrained whole number as specified in section 10.5 of the X.691 standard.
long	`decodeConsWholeNumber(long rangeValue,java.lang.String ident)` This method implements the rules to decode a constrained whole number as specified in section 10.5 of the X.691 standard.
long	`decodeExtLength()` This method decodes an extension length value.
long	`decodeInt(int nocts,boolean signExtend)` This method implements the rules to decode an unconstrained integer value.
long	`decodeInt(int nocts,boolean signExtend,java.lang.String ident)` This method implements the rules to decode an unconstrained integer value.

long	`decodeLength()` This method decodes either a constrained or unconstrained length depending on whether a size constraint object is set in this object.
long	`decodeLength(long lower,long upper)` This method decodes a constrained length determinant value.
int	`decodeSmallLength()` This method implements the rules to decode a normally small length as specified in section 11.9 of the X.691 standard.
int	`decodeSmallNonNegWholeNumber()` This method implements the rules to decode a small non-negative whole number as specified in section 10.6 of the X.691 standard.
long	`decodeUnconsLength()` This method decodes a general (unconstrained) length determinant value as described in section 10.9 of the 2008 X.691 standard.
boolean	`isAligned()` This method tests if PER alignment is turned on or off.
void	`setAligned(boolean value)` This method is used to turn PER aligned encoding on or off.
static Asn1PerDecodeBuffer	`setBuffer(Asn1PerDecodeBuffer buffer,byte[] msgdata,boolean aligned)` This method will create or reinitialize a PER decode message buffer object to read data from the given byte array.
void	`setSizeConstraint(long lower,long upper)` This method is used to set a size constraint within the buffer object.
void	`setSizeConstraintExt(long lower,long upper,long extLower,long extUpper)` This method is used to set an extensible size constraint within the buffer object.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`

`binDump, binDump, byteAlign, decodeBit, decodeBitsToInt, decodeBitsToLong, decodeBitsToOctetArray, decodeBitsToOctetArray, getAvailableBits, getBitOffset, getBitOffsetInByte, getMsgBitCnt, getTraceHandler, hasMoreBits, mark, moveBitCursor, readByte, reset, setInputStream, skipBits`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`

`addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBuffer`

`addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`

`getContext, hexDump, hexDump, setKey, setTypeCode`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1PerDecodeBuffer

```
public Asn1PerDecodeBuffer(byte[] msgdata,
                           boolean aligned)
```

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

Parameters:

`msgdata` - Byte array containing an encoded ASN.1 message.
`aligned` - Boolean specifying PER aligned or unaligned encoding.

Asn1PerDecodeBuffer

```
public Asn1PerDecodeBuffer(java.io.InputStream istream,
                           boolean aligned)
```

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

Parameters:

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

Methods

byteAlign

```
public void byteAlign()
```

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

decodeBit

```
public boolean decodeBit(java.lang.String ident)
  throws Asn1EndOfBufferException,
         java.io.IOException
```

This method decodes a single bit value.

Parameters:

`ident` - Bit field identifier name for tracing.

Returns:

Boolean value of bit that was decoded.

(continued from last page)

decodeBit

```
public boolean decodeBit()
  throws Asn1EndOfBufferException,
         java.io.IOException
```

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

Returns:

Boolean value of bit that was decoded.

---

decodeBitsToLong

```
public long decodeBitsToLong(int nbits,
         java.lang.String ident)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nbits` - Number of bits to decode

`ident` - Bit field identifier name for tracing.

Returns:

Long integer value containing decoded bits

---

decodeBitsToLong

```
public long decodeBitsToLong(int nbits)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nbits` - Number of bits to decode

Returns:

Long integer value containing decoded bits

---

decodeBitsToInt

```
public int decodeBitsToInt(int nbits,
         java.lang.String ident)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nbits` - Number of bits to decode

`ident` - Bit field identifier name for tracing.

Returns:

Integer value containing decoded bits

(continued from last page)

decodeBitsToInt

```
public int decodeBitsToInt(int nbits)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nbits` - Number of bits to decode

Returns:

Integer value containing decoded bits

---

decodeBitsToOctetArray

```
public void decodeBitsToOctetArray(byte[] value,
          int offset,
          int nbits,
          java.lang.String ident)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`value` - Octet array for decoded data
`offset` - Starting byte offset into array
`nbits` - Number of bits to decode
`ident` - Bit field identifier name for tracing, or null.

---

decodeBitsToOctetArray

```
public void decodeBitsToOctetArray(byte[] value,
          int offset,
          int bitOffset,
          int nbits,
          java.lang.String ident)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes bits from the input stream into an array of octets. The user is expected to have provided an array large enough to hold the number of bits requested to be decoded. The first bit is decoded into the given offset byte at the MSB if bitOffset == 0, and at the LSB if bitOffset == 7.

Parameters:

`value` - Octet array for decoded data
`offset` - Starting byte offset into array
`bitOffset` - Where in first byte the first bit goes
`nbits` - Number of bits to decode
`ident` - Bit field identifier name for tracing, or null.

---

decodeBitsToOctetArray

```
public void decodeBitsToOctetArray(byte[] value,
          int offset,
          int nbits)
  throws Asn1Exception,
         java.io.IOException
```

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

(continued from last page)

Parameters:

`value` - Octet array for decoded data
`offset` - Starting byte offset into array
`nbits` - Number of bits to decode

---

decodeCharString

```
public void decodeCharString(int nchars,
         int abpc,
         int ubpc,
         Asn1CharSet charSet,
         java.lang.StringBuffer sbuf)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nchars` - Number of characters
`abpc` - Number of bits per character (aligned)
`ubpc` - Number of bits per character (unaligned)
`charSet` - Object representing the permitted alphabet constraint character set (optional)
`sbuf` - String buffer to receive decoded result

---

decodeConsWholeNumber

```
public long decodeConsWholeNumber(long rangeValue,
         java.lang.String ident)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`rangeValue` - upper - lower + 1. If a negative value is passed, it will be interpreted as an unsigned integer. If 0 is given, it will be interpreted as 2^64.
`ident` - Tracing identifier

Returns:

Decoded adjusted value = value - lower range endpoint value.

---

decodeConsWholeNumber

```
public long decodeConsWholeNumber(long rangeValue)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`rangeValue` - upper - lower + 1. If a negative value is passed, it will be interpreted as an unsigned integer. If 0 is given, it will be interpreted as 2^64.

Returns:

Decoded adjusted value = value - lower range endpoint value

(continued from last page)

decodeExtLength

```
public long decodeExtLength()
  throws Asn1Exception,
         java.io.IOException
```

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

Returns:

Bit offset to next element in buffer

---

decodeInt

```
public long decodeInt(int nocts,
          boolean signExtend,
          java.lang.String ident)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nocts` - Number of octets to decode
`signExtend` - Sign extend resulting value?
`ident` - Tracing identifier

Returns:

Decoded long integer value

---

decodeInt

```
public long decodeInt(int nocts,
          boolean signExtend)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`nocts` - Number of octets to decode
`signExtend` - Sign extend resulting value?

Returns:

Decoded long integer value

---

decodeLength

```
public long decodeLength()
  throws Asn1Exception,
         java.io.IOException
```

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

Returns:

Decoded length value.

(continued from last page)

decodeUnconsLength

```
public long decodeUnconsLength()
  throws Asn1Exception,
         java.io.IOException
```

This method decodes a general (unconstrained) length determinant value as described in section 10.9 of the 2008 X.691 standard. The maximum value that will be returned is 64K which is the largest length fragment size defined for PER. If a value of 16K or larger is returned, the user must repeat this call to fully decode the fragmented contents.

Returns:

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

---

decodeLength

```
public long decodeLength(long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes a constrained length determinant value. If upper >= 64K, it is possible that the data has been fragmented. If this is the case and this method returns >= 16K, then there will be another length encoded following the current chunk.

Parameters:

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

Returns:

Decoded length value

---

decodeSmallLength

```
public int decodeSmallLength()
  throws Asn1Exception,
         java.io.IOException
```

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

---

decodeSmallNonNegWholeNumber

```
public int decodeSmallNonNegWholeNumber()
  throws Asn1Exception,
         java.io.IOException
```

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

---

isAligned

```
public boolean isAligned()
```

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

---

setAligned

```
public void setAligned(boolean value)
```

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

Parameters:

`value` - Boolean specifying PER aligned (true) or unaligned encoding (false).

(continued from last page)

setBuffer

```
public static Asn1PerDecodeBuffer setBuffer(Asn1PerDecodeBuffer buffer,
          byte[] msgdata,
          boolean aligned)
```

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

Parameters:

`buffer` - Existing message buffer object
`msgdata` - Byte array containing message data
`aligned` - Boolean specifying PER aligned or unaligned encoding.

---

setSizeConstraint

```
public void setSizeConstraint(long lower,
          long upper)
```

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

Parameters:

`lower` - Lower bound of root constraint
`upper` - Upper bound of root constraint

---

setSizeConstraintExt

```
public void setSizeConstraintExt(long lower,
          long upper,
          long extLower,
          long extUpper)
```

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

Parameters:

`lower` - Lower bound of root constraint
`upper` - Upper bound of root constraint
`extLower` - Lower bound of extension constraint
`extUpper` - Upper bound of extension constraint

com.objsys.asn1j.runtime
Interface Asn1PerDecoder

public interface **Asn1PerDecoder**

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

Method Summary

`abstract Asn1Type`	`decode(Asn1PerDecodeBuffer buffer)` Decode value from given buffer.

Methods

decode

```
public abstract Asn1Type decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

Decode value from given buffer.

Parameters:

`buffer`

Returns:

com.objsys.asn1j.runtime
Class Asn1PerDecodeTraceHandler

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerTraceHandler
      |
      +-com.objsys.asn1j.runtime.Asn1PerDecodeTraceHandler
```

public class **Asn1PerDecodeTraceHandler**
extends Asn1PerTraceHandler

This is a utility class for handling the collection and printing of bit field trace information. An object of the class is present within the Asn1DecodeBitBuffer class. It is accessed using the 'getTraceHandler' method from within objects of that class. (This was originally for PER but is now used more generally.)

Fields inherited from class `com.objsys.asn1j.runtime.Asn1PerTraceHandler`
`mBitFieldList`

Constructor Summary

public	`Asn1PerDecodeTraceHandler(Asn1DecodeBitBuffer messageBuffer)`

Method Summary

void	`enable()` This method is used to turn PER bit tracing on
void	`print(java.io.PrintStream out,java.lang.String varName)` This method prints the trace to the given output stream in a default format.
void	`reset()` This method resets the trace bit field list.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1PerTraceHandler`
`addElemName, enable, getBitFieldList, newBitField, print, removeLastElemName, reset, setBitCount, setBitOffset, updateBitField`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1PerDecodeTraceHandler

```
public Asn1PerDecodeTraceHandler(Asn1DecodeBitBuffer messageBuffer)
```

Methods

enable

```
public void enable()
```

This method is used to turn PER bit tracing on

---

print

```
public void print(java.io.PrintStream out,
          java.lang.String varName)
```

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

Parameters:

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

---

reset

```
public void reset()
```

This method resets the trace bit field list.

com.objsys.asn1j.runtime
Class Asn1PerEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeBitBuffer
               ┗-com.objsys.asn1j.runtime.Asn1PerEncodeBuffer
```

All Implemented Interfaces:
Asn1PerEncoder, Asn1PerMessageBuffer, Asn1BitMessageBuffer

---

public class **Asn1PerEncodeBuffer**
extends Asn1EncodeBitBuffer
implements Asn1BitMessageBuffer, Asn1PerMessageBuffer, Asn1PerEncoder

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

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeBitBuffer
mByteIndex, mData, mTraceHandler

Fields inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
INITIAL_SIZE

Fields inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
context, mTypeCode

Constructor Summary

public	Asn1PerEncodeBuffer(boolean aligned) This constructor creates a PER encode buffer object with the default initial size.
public	Asn1PerEncodeBuffer(boolean aligned,int size) This constructor creates a PER encode buffer object with the given initial size.

Method Summary

void	byteAlign() This method byte-aligns the buffer if the buffer is set to be aligned.

void	`encodeBit(boolean value)` This method encodes a single bit value.
void	`encodeBit(boolean value,java.lang.String ident)` This method encodes a single bit value.
void	`encodeBits(byte[] value,int offset,int nbits)` This method encodes bit values from an array of octets.
void	`encodeBits(byte[] value,int offset,int bitOffset,int nbits)` This method encodes bit values from an array of octets.
void	`encodeBits(byte[] value,int offset,int bitOffset,int nbits,java.lang.String ident)` This method encodes bit values from an array of octets.
void	`encodeBits(byte[] value,int offset,int nbits,java.lang.String ident)` This method encodes bit values from an array of octets.
void	`encodeCharString(java.lang.String value,int nchars,int offset,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes the contents of a known-multiplier character string type.
void	`encodeConsWholeNumber(long adjustedValue,long rangeValue)` This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.
void	`encodeConsWholeNumber(long adjustedValue,long rangeValue,java.lang.String ident)` This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.
void	`encodeInt(long value,boolean encodeLen,boolean signExtend)` This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.
void	`encodeInt(long value,boolean encodeLen,boolean signExtend,java.lang.String ident)` This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.
void	`encodeInt(long value,int nbits)` This method encodes bit values from an integer value.
void	`encodeInt(long value,int nbits,java.lang.String ident)` This method encodes bit values from an integer value.
long	`encodeLength(long value)` This method encodes either a constrained or unconstrained length depending on whether a size constraint object is set in this object.
void	`encodeLength(long value,long lower,long upper)` This method encodes a constrained length determinant value.

void	`encodeLengthEOM(long value)` This method checks to see if a zero byte needs to be added after a fragmented length has been encoded.
void	`encodeOctetString(byte[] value,int offset,int nbytes)` This method encodes the given array of bytes as an unconstrained octet string value.
void	`encodeOIDLengthAndValue(int[] value)` This method encodes the length and contents of an object identifier value.
void	`encodeOpenType(Asn1PerEncodeBuffer buffer,java.lang.String elemName)` This overloaded version of encodeOpenType will encode the componet in the given PER encode buffer into this PER encode buffer.
void	`encodeOpenType(byte[] value,int offset,int nbytes)` This method encodes the given array of bytes as an open type.
void	`encodeRelOIDLengthAndValue(int[] value)` This method encodes the length and contents of a relative object identifier value.
void	`encodeSmallLength(int value)` This method implements the rules to encode a normally small length as specified in section 11.9 of the X.691 standard.
void	`encodeSmallNonNegWholeNumber(int value)` This method implements the rules to encode a small non-negative whole number as specified in section 10.6 of the X.691 standard.
long	`encodeUnconsLength(long value)` This method encodes a general (unconstrained) length determinant value as described in section 10.9 or the X.691 standard.
boolean	`isAligned()` This method is used to test if PER aligned encoding has been specified.
void	`setAligned(boolean value)` This method is used to turn PER aligned encoding on or off
void	`setSizeConstraint(long lower,long upper)` This method is used to set a size constraint within the buffer object.
void	`setSizeConstraintExt(long lower,long upper,long extLower,long extUpper)` This method is used to set an extensible size constraint within the buffer object.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBitBuffer`

`binDump, byteAlign, checkSize, copy, copy, encodeBit, encodeBits, encodeBits, encodeBits, encodeLongBits, encodeLongBits, getBitOffsetInByte, getBuffer, getByteArrayInputStream, getByteIndex, getMsgBitCnt, getMsgByteCnt, getMsgCopy, getMsgLength, getTraceHandler, hexDump, isByteAligned, reset, reverseBytes, setMsgBitCnt, toString, write`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`

binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1PerEncodeBuffer

```
public Asn1PerEncodeBuffer(boolean aligned)
```

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

Parameters:

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

Asn1PerEncodeBuffer

```
public Asn1PerEncodeBuffer(boolean aligned,
                           int size)
```

This constructor creates a PER encode buffer object with the given initial size. Whenever the buffer becomes full, it will be expanded. For best performance, this size should be large enough to prevent resizing in normal operation.

Parameters:

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

`size` - The initial size in bytes of an encode buffer.

Methods

byteAlign

```
public void byteAlign()
```

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

encodeBit

```
public void encodeBit(boolean value,
         java.lang.String ident)
```

(continued from last page)

This method encodes a single bit value.

Parameters:

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

---

encodeBit

```
public void encodeBit(boolean value)
```

This method encodes a single bit value. This overrides the parent class method in order to default the name for tracing to "value". Equivalent to encodeBit(value, "value");

Parameters:

`value` - Boolean value of bit to be encoded.

---

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int nbits,
         java.lang.String ident)
  throws Asn1InvalidArgException
```

This method encodes bit values from an array of octets.

Parameters:

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

---

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int bitOffset,
         int nbits,
         java.lang.String ident)
  throws Asn1InvalidArgException
```

This method encodes bit values from an array of octets.

Parameters:

`value` - Octet array containing bits to be encoded
`offset` - Starting byte offset in value
`bitOffset` - Starting bit offset in first byte. Pass 0 to begin encoding with the most significant bit. Pass 7 to begin with the least significant bit.
`nbits` - Number of bits to encode
`ident` - Bit field identifier name for tracing, or null.

---

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int bitOffset,
         int nbits)
```

This method encodes bit values from an array of octets. This overrides the parent class method in order to default the name for tracing to "value". Equivalent to: encodeBits(value, offset, bitOffset, nbits, "value");

(continued from last page)

encodeBits

```
public void encodeBits(byte[] value,
          int offset,
          int nbits)
  throws Asn1InvalidArgException
```

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

Parameters:

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

---

encodeCharString

```
public void encodeCharString(java.lang.String value,
          int nchars,
          int offset,
          int abpc,
          int ubpc,
          Asn1CharSet charSet)
  throws Asn1Exception
```

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

Parameters:

`value` - String containing characters to encode
`nchars` - Number of characters from string to encode
`offset` - Offset to first char in string to encode
`abpc` - Number of bits per character (aligned)
`ubpc` - Number of bits per character (unaligned)
`charSet` - Object representing permitted alphabet constraint character set (optional)

---

encodeConsWholeNumber

```
public void encodeConsWholeNumber(long adjustedValue,
          long rangeValue,
          java.lang.String ident)
  throws Asn1InvalidArgException
```

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

Parameters:

`adjustedValue` - Adjusted value to be encoded = value - lower range endpoint value. Negative values are interpreted as unsigned integers.
`rangeValue` - upper - lower + 1. Negative values are interpreted as unsigned integers. Zero is used to represent 2^64.
`ident` - Tracing identifier

---

encodeConsWholeNumber

```
public void encodeConsWholeNumber(long adjustedValue,
          long rangeValue)
  throws Asn1InvalidArgException
```

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

Parameters:

`adjustedValue` - Adjusted value to be encoded = value - lower range endpoint value
`rangeValue` - upper - lower + 1. Negative values are interpreted as unsigned integers. Zero is used to represent 2^64.

encodeInt

```
public void encodeInt(long value,
         int nbits,
         java.lang.String ident)
  throws Asn1InvalidArgException
```

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

Parameters:

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

encodeInt

```
public void encodeInt(long value,
         int nbits)
  throws Asn1InvalidArgException
```

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

Parameters:

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

encodeInt

```
public void encodeInt(long value,
         boolean encodeLen,
         boolean signExtend,
         java.lang.String ident)
  throws Asn1InvalidArgException
```

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

Parameters:

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

encodeInt

```
public void encodeInt(long value,
         boolean encodeLen,
         boolean signExtend)
  throws Asn1InvalidArgException
```

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

Parameters:

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

(continued from last page)

encodeLength

```
public long encodeLength(long value)
  throws Asn1InvalidArgException
```

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

Parameters:

`value` - Length value to be encoded

Returns:

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

---

encodeUnconsLength

```
public long encodeUnconsLength(long value)
  throws Asn1InvalidArgException
```

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

Parameters:

`value` - Length value to be encoded

Returns:

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

---

encodeLength

```
public void encodeLength(long value,
          long lower,
          long upper)
  throws Asn1Exception
```

This method encodes a constrained length determinant value. This method should not be used when fragmentation is called for (ie, upper >= 64K and value >= 16K). See encodeLength(value) instead.

Parameters:

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

---

encodeLengthEOM

```
public void encodeLengthEOM(long value)
```

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

Parameters:

`value` - Original length value that was encoded.

---

encodeOIDLengthAndValue

```
public void encodeOIDLengthAndValue(int[] value)
  throws Asn1Exception
```

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

Parameters:

`value` - Integer array containing arcs to encode

encodeRelOIDLengthAndValue

```
public void encodeRelOIDLengthAndValue(int[] value)
  throws Asn1Exception
```

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

Parameters:

`value` - Integer array containing arcs to encode

encodeOctetString

```
public void encodeOctetString(byte[] value,
         int offset,
         int nbytes)
  throws Asn1Exception
```

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

Parameters:

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

encodeOpenType

```
public void encodeOpenType(byte[] value,
         int offset,
         int nbytes)
  throws Asn1Exception
```

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

Parameters:

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

encodeOpenType

```
public void encodeOpenType(Asn1PerEncodeBuffer buffer,
         java.lang.String elemName)
  throws Asn1Exception
```

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

Parameters:

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

encodeSmallLength

```
public void encodeSmallLength(int value)
  throws Asn1InvalidArgException
```

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

Parameters:

`value` - Value to be encoded

(continued from last page)

encodeSmallNonNegWholeNumber

```
public void encodeSmallNonNegWholeNumber(int value)
  throws Asn1InvalidArgException
```

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

Parameters:

`value` - Value to be encoded

---

isAligned

```
public boolean isAligned()
```

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

---

setAligned

```
public void setAligned(boolean value)
```

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

---

setSizeConstraint

```
public void setSizeConstraint(long lower,
         long upper)
```

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

Parameters:

`lower` - Lower bound of root constraint

`upper` - Upper bound of root constraint

---

setSizeConstraintExt

```
public void setSizeConstraintExt(long lower,
         long upper,
         long extLower,
         long extUpper)
```

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

Parameters:

`lower` - Lower bound of root constraint

`upper` - Upper bound of root constraint

`extLower` - Lower bound of extension constraint

`extUpper` - Upper bound of extension constraint

com.objsys.asn1j.runtime
Interface Asn1PerEncoder

All Known Implementing Classes:
Asn1PerOutputStream, Asn1PerEncodeBuffer

---

public interface **Asn1PerEncoder**

Common interface for PER encoding methods, implemented by Asn1PerEncodeBuffer and Asn1PerOutputStream.

Method Summary

abstract void	`encodeBit(boolean value)` This method encodes a single bit value.
abstract void	`encodeBit(boolean value,java.lang.String ident)` This method encodes a single bit value.
abstract void	`encodeBits(byte[] value,int offset,int nbits)` This method encodes bit values from an array of octets.
abstract void	`encodeBits(byte[] value,int offset,int nbits,java.lang.String ident)` This method encodes bit values from an array of octets.
abstract void	`encodeBits(byte value,int nbits)` This method encodes bit values from an octet.
abstract void	`encodeCharString(java.lang.String value,int nchars,int offset,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes the contents of a known-multiplier character string type.
abstract void	`encodeConsWholeNumber(long adjustedValue,long rangeValue)` This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.
abstract void	`encodeConsWholeNumber(long adjustedValue,long rangeValue,java.lang.String ident)` This method implements the rules to encode a constrained whole number as specified in section 10.5 of the X.691 standard.
abstract void	`encodeInt(long value,boolean encodeLen,boolean signExtend)` This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.
abstract void	`encodeInt(long value,boolean encodeLen,boolean signExtend,java.lang.String ident)` This method implements the rules to encode either a non-negative binary integer as specified in section 10.3 or a two's complement binary integer as specified in section 10.4 of the X.691 standard.

abstract void	encodeInt(long value,int nbits) This method encodes bit values from an integer value.
abstract void	encodeInt(long value,int nbits,java.lang.String ident) This method encodes bit values from an integer value.
abstract long	encodeLength(long value) This method encodes a general (unconstrained) length determinant value as described in section 10.9 or the X.691 standard.
abstract void	encodeLength(long value,long lower,long upper) This method encodes a constrained length determinant value.
abstract void	encodeLengthEOM(long value) This method checks to see if a zero byte needs to be added after a fragmented length has been encoded.
abstract void	encodeOctetString(byte[] value,int offset,int nbytes) This method encodes the given array of bytes as an unconstrained octet string value.
abstract void	encodeOIDLengthAndValue(int[] value) This method encodes the length and contents of an object identifier value.
abstract void	encodeOpenType(byte[] value,int offset,int nbytes) This method encodes the given array of bytes as an open type.
abstract void	encodeRelOIDLengthAndValue(int[] value) This method encodes the length and contents of a relative object identifier value.
abstract void	encodeSmallLength(int value) This method implements the rules to encode a normally small length as specified in section 11.9 of the X.691 standard.
abstract void	encodeSmallNonNegWholeNumber(int value) This method implements the rules to encode a small non-negative whole number as specified in section 10.6 of the X.691 standard.

Methods

encodeBit

```
public abstract void encodeBit(boolean value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes a single bit value.

Parameters:

`value` - Boolean value of bit to be encoded.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

encodeBit

```
public abstract void encodeBit(boolean value,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes a single bit value.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

encodeBits

```
public abstract void encodeBits(byte value,
         int nbits)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

encodeBits

```
public abstract void encodeBits(byte[] value,
         int offset,
         int nbits)
  throws java.io.IOException,
         Asn1InvalidArgException
```

This method encodes bit values from an array of octets.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

encodeBits

```
public abstract void encodeBits(byte[] value,
         int offset,
         int nbits,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

(continued from last page)

This method encodes bit values from an array of octets.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeCharString

```
public abstract void encodeCharString(java.lang.String value,
         int nchars,
         int offset,
         int abpc,
         int ubpc,
         Asn1CharSet charSet)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - String containing characters to encode
`nchars` - Number of characters from string to encode
`offset` - Offset to first char in string to encode
`abpc` - Number of bits per character (aligned)
`ubpc` - Number of bits per character (unaligned)
`charSet` - Object representing permitted alphabet constraint character set (optional)

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeConsWholeNumber

```
public abstract void encodeConsWholeNumber(long adjustedValue,
         long rangeValue,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

(continued from last page)

encodeConsWholeNumber

```
public abstract void encodeConsWholeNumber(long adjustedValue,
         long rangeValue)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

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

`rangeValue` - upper - lower + 1. Negative values are interpreted as unsigned integers. Zero is used to represent 2^64.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeInt

```
public abstract void encodeInt(long value,
         int nbits,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

`value` - Integer containing bits to be encoded

`nbits` - Number of bits to encode

`ident` - Bit field identifier name for tracing.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeInt

```
public abstract void encodeInt(long value,
         int nbits)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

`value` - Integer containing bits to be encoded

`nbits` - Number of bits to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeInt

```
public abstract void encodeInt(long value,
         boolean encodeLen,
         boolean signExtend,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

(continued from last page)

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeInt

```
public abstract void encodeInt(long value,
         boolean encodeLen,
         boolean signExtend)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeLength

```
public abstract long encodeLength(long value)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

`value` - Length value to be encoded

Returns:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeLength

```
public abstract void encodeLength(long value,
         long lower,
         long upper)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes a constrained length determinant value.

(continued from last page)

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeLengthEOM

```
public abstract void encodeLengthEOM(long value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Original length value that was encoded.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeOctetString

```
public abstract void encodeOctetString(byte[] value,
          int offset,
          int nbytes)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeOIDLengthAndValue

```
public abstract void encodeOIDLengthAndValue(int[] value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Integer array containing arcs to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

(continued from last page)

encodeRelOIDLengthAndValue

```
public abstract void encodeRelOIDLengthAndValue(int[] value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Integer array containing arcs to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeOpenType

```
public abstract void encodeOpenType(byte[] value,
          int offset,
          int nbytes)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeSmallLength

```
public abstract void encodeSmallLength(int value)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

`value` - Value to be encoded

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

---

encodeSmallNonNegWholeNumber

```
public abstract void encodeSmallNonNegWholeNumber(int value)
  throws java.io.IOException,
         Asn1InvalidArgException
```

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

Parameters:

`value` - Value to be encoded

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1InvalidArgException` - Any exception thrown by the underlying Asn1PerEncodeBuffer.

com.objsys.asn1j.runtime
Class Asn1PerEncodeTraceHandler

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerTraceHandler
      |
      +-com.objsys.asn1j.runtime.Asn1PerEncodeTraceHandler
```

public class **Asn1PerEncodeTraceHandler**
extends Asn1PerTraceHandler

This is a utility class for handling the collection and printing of bit field trace information. An object of the class is present within the Asn1EncodeBitBuffer class. It is accessed using the 'getTraceHandler' method from within objects of that class. (This was originally for PER, but is now used more generally.)

Fields inherited from class `com.objsys.asn1j.runtime.Asn1PerTraceHandler`
`mBitFieldList`

Constructor Summary

public	`Asn1PerEncodeTraceHandler(Asn1EncodeBitBuffer messageBuffer)` This constructor initializes internal trace handler member variables.

Method Summary

void	`enable()` This method is used to turn PER bit tracing on
void	`print(java.io.PrintStream out,java.lang.String varName)` This method prints the trace to the given output stream in a default format.
void	`reset()` This method resets the trace bit field list.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1PerTraceHandler`
`addElemName, enable, getBitFieldList, newBitField, print, removeLastElemName, reset, setBitCount, setBitOffset, updateBitField`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1PerEncodeTraceHandler

```
public Asn1PerEncodeTraceHandler(Asn1EncodeBitBuffer messageBuffer)
```

This constructor initializes internal trace handler member variables.

Parameters:

`messageBuffer` - PER encode message buffer object reference

Methods

enable

```
public void enable()
```

This method is used to turn PER bit tracing on

---

print

```
public void print(java.io.PrintStream out,
         java.lang.String varName)
```

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

Parameters:

`out` - Print stream to which output is to be written.

`varName` - Name of the object variable being printed.

---

reset

```
public void reset()
```

This method resets the trace bit field list.

com.objsys.asn1j.runtime
Class Asn1PerInputStream

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1DecodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1DecodeBitBuffer
               ┗-com.objsys.asn1j.runtime.Asn1PerDecodeBuffer
                  ┗-com.objsys.asn1j.runtime.Asn1PerInputStream
```

All Implemented Interfaces:
Asn1InputStream, Asn1BitMessageBuffer, Asn1PerMessageBuffer

---

public class **Asn1PerInputStream**
extends Asn1PerDecodeBuffer
implements Asn1PerMessageBuffer, Asn1BitMessageBuffer, Asn1InputStream

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBitBuffer`
`mTraceHandler`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1DecodeBuffer`
`mByteCount`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

`public`	`Asn1PerInputStream(java.io.InputStream istream,boolean aligned)` This constructor creates a PER input stream object that references an encoded ASN.1 message.

Method Summary

`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.

void	close() Closes this input stream and releases any system resources associated with the stream.
void	mark(int readLimit) This method is used to mark the current position in the input stream for retry processing.
boolean	markSupported() Tests if this input stream supports the markand resetmethods.
void	reset() This method is used to reset the current position in the input stream back to the location of the last 'mark' call.
long	skip(long nbytes) This method will skip over the requested number of bytes in the input stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1PerDecodeBuffer

byteAlign, decodeBit, decodeBit, decodeBitsToInt, decodeBitsToInt, decodeBitsToLong, decodeBitsToLong, decodeBitsToOctetArray, decodeBitsToOctetArray, decodeBitsToOctetArray, decodeCharString, decodeConsWholeNumber, decodeConsWholeNumber, decodeExtLength, decodeInt, decodeInt, decodeLength, decodeLength, decodeSmallLength, decodeSmallNonNegWholeNumber, decodeUnconsLength, isAligned, setAligned, setBuffer, setSizeConstraint, setSizeConstraintExt

Methods inherited from class com.objsys.asn1j.runtime.Asn1DecodeBitBuffer

binDump, binDump, byteAlign, decodeBit, decodeBitsToInt, decodeBitsToLong, decodeBitsToOctetArray, decodeBitsToOctetArray, getAvailableBits, getBitOffset, getBitOffsetInByte, getMsgBitCnt, getTraceHandler, hasMoreBits, mark, moveBitCursor, readByte, reset, setInputStream, skipBits

Methods inherited from class com.objsys.asn1j.runtime.Asn1DecodeBuffer

addCaptureBuffer, capture, decodeIntValue, decodeOIDContents, decodeRelOIDContents, getByteCount, getInputStream, getLazyOpenTypeDecode, hexDump, init, mark, read, read, read, read2Bytes, read4Bytes, readByte, removeCaptureBuffer, reset, setInputStream, setLazyOpenTypeDecode, skip

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer

addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase

getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

(continued from last page)

Constructors

Asn1PerInputStream

```
public Asn1PerInputStream(java.io.InputStream istream,
                          boolean aligned)
```

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

Parameters:

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

Methods

available

```
public int available()
  throws java.io.IOException
```

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.

Throws:

`IOException` - if an I/O error occurs.

---

close

```
public void close()
  throws java.io.IOException
```

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

Throws:

`IOException` - if an I/O error occurs.

---

mark

```
public void mark(int readLimit)
```

This method is used to mark the current position in the input stream for retry processing. It is equivalent to the Java InputStream 'mark' method.

Parameters:

`readLimit` - Max number of bytes that can be read before mark becomes invalid.

---

markSupported

```
public boolean markSupported()
```

Tests if this input stream supports the `mark`and `reset`methods. The `markSupported`method of `InputStream`returns `false`.

Returns:

`true`if this true type supports the mark and reset method; `false`otherwise.

(continued from last page)

reset

```
public 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 the Java InputStream 'reset' method.

---

skip

```
public long skip(long nbytes)
  throws java.io.IOException
```

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

Parameters:

`nbytes` - Number of bytes to skip

Throws:

`IOException` - if an I/O error occurs.

com.objsys.asn1j.runtime
Interface Asn1PerMessageBuffer

All Superinterfaces:
Asn1BitMessageBuffer

All Known Implementing Classes:
Asn1PerEncodeBuffer, Asn1PerDecodeBuffer

---

public interface **Asn1PerMessageBuffer**
extends Asn1BitMessageBuffer

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

Method Summary

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

Methods inherited from interface com.objsys.asn1j.runtime.Asn1BitMessageBuffer
byteAlign, getInputStream, getMsgBitCnt, getTraceHandler

Methods

isAligned

public abstract boolean **isAligned**()

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

com.objsys.asn1j.runtime
Class Asn1PerOutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-com.objsys.asn1j.runtime.Asn1OutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1PerOutputStream
```

All Implemented Interfaces:
Asn1PerEncoder, java.io.Flushable, java.io.Closeable

---

public class **Asn1PerOutputStream**
extends Asn1OutputStream
implements java.io.Closeable, java.io.Flushable, Asn1PerEncoder

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

Field Summary

Asn1PerOutputStreamTraceHandler	mTraceHandler

Fields inherited from class com.objsys.asn1j.runtime.Asn1OutputStream
os

Constructor Summary

public	Asn1PerOutputStream(java.io.OutputStream os,boolean aligned) This constructor creates a buffered PER output stream object with the default size of buffer.
public	Asn1PerOutputStream(java.io.OutputStream os,int bufSize,boolean aligned) This constructor creates a buffered PER output stream object with the specified size of buffer.

Method Summary

void	addCaptureBuffer(java.io.ByteArrayOutputStream buffer) This method is used to add a capture buffer to the internal capture buffer list.
void	binDump(java.io.PrintStream out,java.lang.String varName) This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.

void	binDump(java.lang.String varName) This method invokes an overloaded version of binDump to dump the encoded message to standard output.
void	byteAlign() This methods byte-aligns the buffer.
void	close() Close the stream.
void	encodeBit(boolean value)
void	encodeBit(boolean value,java.lang.String ident)
void	encodeBits(byte[] value,int offset,int nbits)
void	encodeBits(byte[] value,int offset,int nbits,java.lang.String ident)
void	encodeBits(byte value,int nbits)
void	encodeCharString(java.lang.String value,int nchars,int offset,int abpc,int ubpc,Asn1CharSet charSet)
void	encodeConsWholeNumber(long adjustedValue,long rangeValue)
void	encodeConsWholeNumber(long adjustedValue,long rangeValue,java.lang.String ident)
void	encodeInt(long value,boolean encodeLen,boolean signExtend)
void	encodeInt(long value,boolean encodeLen,boolean signExtend,java.lang.String ident)
void	encodeInt(long value,int nbits)
void	encodeInt(long value,int nbits,java.lang.String ident)
long	encodeLength(long value)
void	encodeLength(long value,long lower,long upper)
void	encodeLengthEOM(long value)
void	encodeOctetString(byte[] value,int offset,int nbytes)
void	encodeOIDLengthAndValue(int[] value)
void	encodeOpenType(byte[] value,int offset,int nbytes)
void	encodeRelOIDLengthAndValue(int[] value)

void	encodeSmallLength(int value)
void	encodeSmallNonNegWholeNumber(int value)
void	flush() Flush the buffer to the stream.
Asn1PerTraceHandler	getTraceHandler() This method will return a reference to the internal trace handler object used to trace the bit fields within a PER message.
boolean	isAligned() This method is used to test if PER aligned encoding has been specified.
void	removeCaptureBuffer(java.io.ByteArrayOutputStream buffer) This method is used to remove a capture buffer from the internal capture buffer list.
void	write(byte[] b) Writes b.lengthbytes from the specified byte array to this output stream.
void	write(byte[] b,int off,int len) Writes lenbytes from the specified byte array starting at offset offto this output stream.
void	write(int b) Writes the specified byte to this output stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1OutputStream
close, flush, getContext, write, write, write, write2Bytes, write4Bytes

Methods inherited from class java.io.OutputStream
close, flush, write, write, write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mTraceHandler

protected com.objsys.asn1j.runtime.Asn1PerOutputStreamTraceHandler **mTraceHandler**

Constructors

(continued from last page)

Asn1PerOutputStream

```
public Asn1PerOutputStream(java.io.OutputStream os,
                           boolean aligned)
```

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

Parameters:

`os` - The underlying OutputStream object.
`aligned` - Indicates whether PER aligned or unaligned encoding should be done.

Asn1PerOutputStream

```
public Asn1PerOutputStream(java.io.OutputStream os,
                           int bufSize,
                           boolean aligned)
```

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

Parameters:

`os` - The underlying OutputStream object.
`bufSize` - The buffer size.
`aligned` - Indicates whether PER aligned or unaligned encoding should be done.

Methods

getTraceHandler

```
public Asn1PerTraceHandler getTraceHandler()
```

This method will return a reference to the internal trace handler object used to trace the bit fields within a PER message.

---

binDump

```
public void binDump(java.lang.String varName)
```

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

---

binDump

```
public void binDump(java.io.PrintStream out,
         java.lang.String varName)
```

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

---

flush

```
public void flush()
  throws java.io.IOException
```

Flush the buffer to the stream. It writes whole bytes only.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

close

```
public void close()
  throws java.io.IOException
```

Close the stream. Writes all bytes (even unfinished) before closing.

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

---

isAligned

```
public boolean isAligned()
```

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

---

byteAlign

```
public void byteAlign()
```

This methods byte-aligns the buffer.

---

encodeBit

```
public void encodeBit(boolean value)
  throws java.io.IOException,
         Asn1Exception
```

---

encodeBit

```
public void encodeBit(boolean value,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1Exception
```

---

encodeBits

```
public void encodeBits(byte value,
         int nbits)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int nbits)
  throws java.io.IOException,
         Asn1InvalidArgException
```

(continued from last page)

encodeBits

```
public void encodeBits(byte[] value,
         int offset,
         int nbits,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeCharString

```
public void encodeCharString(java.lang.String value,
         int nchars,
         int offset,
         int abpc,
         int ubpc,
         Asn1CharSet charSet)
  throws java.io.IOException,
         Asn1Exception
```

---

encodeConsWholeNumber

```
public void encodeConsWholeNumber(long adjustedValue,
         long rangeValue,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeConsWholeNumber

```
public void encodeConsWholeNumber(long adjustedValue,
         long rangeValue)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeInt

```
public void encodeInt(long value,
         int nbits,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeInt

```
public void encodeInt(long value,
         int nbits)
  throws java.io.IOException,
         Asn1InvalidArgException
```

(continued from last page)

encodeInt

```
public void encodeInt(long value,
         boolean encodeLen,
         boolean signExtend,
         java.lang.String ident)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeInt

```
public void encodeInt(long value,
         boolean encodeLen,
         boolean signExtend)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeLength

```
public long encodeLength(long value)
  throws java.io.IOException,
         Asn1InvalidArgException
```

---

encodeLength

```
public void encodeLength(long value,
         long lower,
         long upper)
  throws java.io.IOException,
         Asn1Exception
```

---

encodeLengthEOM

```
public void encodeLengthEOM(long value)
  throws java.io.IOException,
         Asn1Exception
```

---

encodeOctetString

```
public void encodeOctetString(byte[] value,
         int offset,
         int nbytes)
  throws java.io.IOException,
         Asn1Exception
```

---

encodeOIDLengthAndValue

```
public void encodeOIDLengthAndValue(int[] value)
  throws java.io.IOException,
         Asn1Exception
```

(continued from last page)

encodeRelOIDLengthAndValue

```
public void encodeRelOIDLengthAndValue(int[] value)
  throws java.io.IOException,
         Asn1Exception
```

encodeOpenType

```
public void encodeOpenType(byte[] value,
         int offset,
         int nbytes)
  throws java.io.IOException,
         Asn1Exception
```

encodeSmallLength

```
public void encodeSmallLength(int value)
  throws java.io.IOException,
         Asn1InvalidArgException
```

encodeSmallNonNegWholeNumber

```
public void encodeSmallNonNegWholeNumber(int value)
  throws java.io.IOException,
         Asn1InvalidArgException
```

addCaptureBuffer

```
public void addCaptureBuffer(java.io.ByteArrayOutputStream buffer)
```

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

Parameters:

`buffer` - Buffer into which captured bytes are to be stored

removeCaptureBuffer

```
public void removeCaptureBuffer(java.io.ByteArrayOutputStream buffer)
```

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

Parameters:

`buffer` - Buffer in which captured bytes stored

write

```
public void write(byte[] b)
  throws java.io.IOException
```

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.

(continued from last page)

Throws:

`IOException` - if an I/O error occurs.

---

write

```
public void write(byte[] b,
                  int off,
                  int len)
  throws java.io.IOException
```

Writes `len`bytes from the specified byte array starting at offset `off`to this output stream.

Parameters:

`b` - the data.

`off` - the start offset in the data.

`len` - the number of bytes to write.

Throws:

`IOException` - if an I/O error occurs. In particular, an `IOException`is thrown if the output stream is closed.

---

write

```
public void write(int b)
  throws java.io.IOException
```

Writes the specified byte to this output stream.

Parameters:

`b` - the `byte`.

Throws:

`IOException` - if an I/O error occurs. In particular, an `IOException`may be thrown if the output stream has been closed.

com.objsys.asn1j.runtime
Class Asn1PerOutputStreamTraceHandler

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerTraceHandler
      |
      +-com.objsys.asn1j.runtime.Asn1PerOutputStreamTraceHandler
```

public class **Asn1PerOutputStreamTraceHandler**
extends Asn1PerTraceHandler

This is a utility class for handling the collection and printing of PER bit field trace information for PER output stream. An object of the class is present in the Asn1PerOutputStream classes. It is accessed using the 'getTraceHandler' method from within objects of these classes.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1PerTraceHandler`
`mBitFieldList`

Constructor Summary

public	`Asn1PerOutputStreamTraceHandler(Asn1PerOutputStream out)`

Method Summary

void	`enable()` This method is used to turn PER bit tracing on
void	`print(java.io.PrintStream out,java.lang.String varName)` This method prints the trace to the given output stream in a default format.
void	`reset()` This method does nothing here.
void	`resetTrace()` This method resets the trace bit field list.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1PerTraceHandler`
`addElemName, enable, getBitFieldList, newBitField, print, removeLastElemName, reset, setBitCount, setBitOffset, updateBitField`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

(continued from last page)

Constructors

Asn1PerOutputStreamTraceHandler

```
public Asn1PerOutputStreamTraceHandler(Asn1PerOutputStream out)
```

Methods

enable

```
public void enable()
```

This method is used to turn PER bit tracing on

---

print

```
public void print(java.io.PrintStream out,
          java.lang.String varName)
```

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

Parameters:

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

---

reset

```
public void reset()
```

This method does nothing here.

---

resetTrace

```
public void resetTrace()
```

This method resets the trace bit field list.

com.objsys.asn1j.runtime
Class Asn1PerTime

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1TimeUtil
      |
      +-com.objsys.asn1j.runtime.Asn1PerTime
```

Deprecated. *Use Asn1TimeUtil instead. All members have been relocated to the Asn1TimeUtil class.*

public class **Asn1PerTime**
extends Asn1TimeUtil

Asn1PerTime exists as a subclass of Asn1TimeUtil simply to preserve the original name (Asn1PerTime) and provide backward-compatibility.

Fields inherited from class com.objsys.asn1j.runtime.Asn1TimeUtil
ANY, CENTURY, DATE, DATE_TIME, DAY, DIFF, DT_ANY_CENTURY, DT_ANY_YEAR, DT_ANY_YEAR_DAY, DT_ANY_YEAR_MONTH, DT_ANY_YEAR_MONTH_DAY, DT_ANY_YEAR_WEEK, DT_ANY_YEAR_WEEK_DAY, DT_CENTURY, DT_HOURS, DT_HOURS_AND_DIFF, DT_HOURS_AND_DIFF_AND_FRACTION, DT_HOURS_AND_FRACTION, DT_HOURS_UTC, DT_HOURS_UTC_AND_FRACTION, DT_MINUTES, DT_MINUTES_AND_DIFF, DT_MINUTES_AND_DIFF_AND_FRACTION, DT_MINUTES_AND_FRACTION, DT_MINUTES_UTC, DT_MINUTES_UTC_AND_FRACTION, DT_SECONDS, DT_SECONDS_AND_DIFF, DT_SECONDS_AND_DIFF_AND_FRACTION, DT_SECONDS_AND_FRACTION, DT_SECONDS_UTC, DT_SECONDS_UTC_AND_FRACTION, DT_YEAR, DT_YEAR_DAY, DT_YEAR_MONTH, DT_YEAR_MONTH_DAY, DT_YEAR_WEEK, DT_YEAR_WEEK_DAY, DURATION, FRACTION, HOURS, MINUTES, MONTH, SECONDS, TIME_OF_DAY, UTC, WEEK, YEAR

Constructor Summary

public	Asn1PerTime() **Deprecated.**

Methods inherited from class com.objsys.asn1j.runtime.Asn1TimeUtil
convertToDER, decodeDate, decodeDate, decodeDateTime, decodeDateTime, decodeDuration, decodeDuration, decodeIntervalDE, decodeIntervalSD, decodeIntervalSE, decodeTime, decodeTime, encodeDate, encodeDate, encodeDateTime, encodeDateTime, encodeDuration, encodeDuration, encodeIntervalDE, encodeIntervalSD, encodeIntervalSE, encodeTime, encodeTime

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

Asn1PerTime

`public` **`Asn1PerTime`**`()`

Deprecated.

com.objsys.asn1j.runtime
Class Asn1PerTraceHandler

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1PerTraceHandler
```

Direct Known Subclasses:
Asn1PerOutputStreamTraceHandler, Asn1PerEncodeTraceHandler, Asn1PerDecodeTraceHandler

---

public abstract class **Asn1PerTraceHandler**
extends java.lang.Object

This is the abstract base class for the encode and decode trace handler derived classes. (This was originally for PER but is now used more generally.)

Field Summary

Asn1PerBitFieldList	mBitFieldList

Constructor Summary

protected	Asn1PerTraceHandler(Asn1BitMessageBuffer messageBuffer) This constructor initializes internal trace handler member variables.

Method Summary

void	addElemName(java.lang.String name,int arrayx) This method adds an element name to the current fully qualified name.
abstract void	enable() This method is used to turn PER bit tracing on or off
Asn1PerBitFieldList	getBitFieldList() This method returns a reference to the bit field list
void	newBitField(java.lang.String name,int bitCount) This method creates a new bit field and appends it to the bit field list.
abstract void	print(java.io.PrintStream out,java.lang.String varName) This method prints the trace to the given output stream in a default format.
void	removeLastElemName() This method removes the last element name int the current fully qualified name string.
abstract void	reset() This method resets the trace bit field list.

void	`setBitCount()` This method sets the bit count within the current bit field to the difference between the current but offset and the starting bit offset currently stored in the field object.
void	`setBitOffset()` This method sets the bit offset within the current bit field to the current offset within the PER message buffer.
void	`updateBitField()` This method updates the length of the previously added bit field based on the current buffer position.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

mBitFieldList

```
protected com.objsys.asn1j.runtime.Asn1PerBitFieldList mBitFieldList
```

Constructors

Asn1PerTraceHandler

```
protected Asn1PerTraceHandler(Asn1BitMessageBuffer messageBuffer)
```

This constructor initializes internal trace handler member variables.

Parameters:

`messageBuffer` - PER message buffer object reference

Methods

addElemName

```
public void addElemName(java.lang.String name,
                        int arrayx)
```

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

Parameters:

`name` - Name component to append to string

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

enable

```
public abstract void enable()
```

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

getBitFieldList

```
public Asn1PerBitFieldList getBitFieldList()
```

This method returns a reference to the bit field list

newBitField

```
public void newBitField(java.lang.String name,
          int bitCount)
```

This method creates a new bit field and appends it to the bit field list. The field begins at the current location and has the given length. If the length is unknown, use zero and then make a subsequent call to updateBitField to update the length.

Parameters:

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

`bitCount` - Number of bits in the bit field.

updateBitField

```
public final void updateBitField()
```

This method updates the length of the previously added bit field based on the current buffer position. This can be invoked just after encoding a field to update that field's length. You must have called newBitField just prior to encoding the field.

print

```
public abstract void print(java.io.PrintStream out,
          java.lang.String varName)
```

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

Parameters:

`out` - Print stream to which output is to be written.

`varName` - Name of the object variable being printed.

removeLastElemName

```
public void removeLastElemName()
```

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

reset

```
public abstract void reset()
```

This method resets the trace bit field list.

setBitCount

```
public void setBitCount()
```

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

setBitOffset

```
public void setBitOffset()
```

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

com.objsys.asn1j.runtime
Class Asn1PrintableString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1PrintableString
```

public class **Asn1PrintableString**
extends Asn18BitCharString

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

Field Summary

static final Asn1CharSet	CHARSET
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 19).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1PrintableString() The default constructor creates an empty string object.
public	Asn1PrintableString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`

`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

CHARSET

`public static final com.objsys.asn1j.runtime.Asn1CharSet` **`CHARSET`**

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 19).

Constructors

(continued from last page)

Asn1PrintableString

```
public Asn1PrintableString()
```

The default constructor creates an empty string object.

Asn1PrintableString

```
public Asn1PrintableString(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

com.objsys.asn1j.runtime
Class Asn1Real

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Real
```

Direct Known Subclasses:
Asn1RealBase2, Asn1BigDecimal

---

public class **Asn1Real**
extends Asn1Type

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

Field Summary

int	ANY_BASE
int	ANY_BINARY
int	BASE_2_ONLY
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 9).
double	value This public member variable is where the double value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Real() The default constructor sets the double value to zero.
public	Asn1Real(double value_) This constructor creates an REAL object from a double value.
public	Asn1Real(boolean floatType) **Deprecated.** *use Asn1Real() or Asn1Real(double) instead.*

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 REAL value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1DecodeBuffer buffer,int length,int baseflag)` This method decodes the content octets of an ASN.1 REAL value into this object, where the REAL was encoded as for BER and the length is taken to be as given.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 REAL from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode a REAL value, encoded according to OER, into this object.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes ASN.1 REAL value using the Packed Encoding Rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,int baseflag)` This method decodes ASN.1 REAL value using the Packed Encoding Rules (PER).
void	`decodeDouble(Asn1OerDecodeBuffer buffer)` Decode a REAL value, encoded according to OER in double precision format.
void	`decodeSingle(Asn1OerDecodeBuffer buffer)` Decode a REAL value, encoded according to OER in single precision format.
void	`decodeV72(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 REAL from JSON.
void	`decodeXER(java.lang.String buffer,java.lang.String attrs,boolean decodingElemName,boolean modifiedEncodings)` This method decodes an ASN.1 real value using XER.
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 real value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 REAL value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this ASN.1 real value to JSON.
void	`encode(Asn1OerEncodeBuffer buffer)` Encode this REAL value, according to OER, into the buffer.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes ASN.1 REAL value using the Packed Encoding Rules (PER).

void	encode(Asn1PerOutputStream out) This method encodes ASN.1 REAL value using the Packed Encoding Rules (PER).
void	encode(Asn1XerEncoder buffer,java.lang.String elemName) This method encodes an ASN.1 real value using the XML encoding rules (XER).
void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an ASN.1 real value according to Obj-Sys encoding rules.
void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix,boolean asText) This method encodes an ASN.1 real value according to XER encoding rules.
void	encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName) This method encodes an ASN.1 real value using the XML Encoding as specified in the W3C XML schema standard(asn2xsd).
void	encodeDouble(Asn1OerEncodeBuffer buffer) Encode this REAL value, according to OER, in double precision format, into the buffer.
void	encodeSingle(Asn1OerEncodeBuffer buffer) Encode this REAL value, according to OER, in single precision format, into the buffer.
void	encodeV72(Asn1JsonOutputStream outstream) Encode this ASN.1 real value to JSON.
void	encodeValue(Asn1XmlEncoder buffer) This method encodes an ASN.1 real value using the XML encoding (non-XER).
boolean	equals(double dvalue) This method compares this REAL value to the given value for equality.
boolean	equals(java.lang.Object o) This method compares this REAL value to the given Object for equality.
java.lang.String	getAsn1TypeName() Returns the ASN.1 type name.
int	hashCode() Returns the hash code for this REAL object.
static java.lang.String	normalizedRealValueToString(double value) This method will return a string representation of the normalized REAL value.
java.lang.String	toString() This method will return a string representation of the REAL value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 9).

value

`public transient double` **`value`**

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.

ANY_BASE

`protected int` **`ANY_BASE`**

ANY_BINARY

`protected int` **`ANY_BINARY`**

BASE_2_ONLY

`protected int` **`BASE_2_ONLY`**

Constructors

Asn1Real

`public` **`Asn1Real`**`()`

The default constructor sets the double value to zero.

Asn1Real

`public` **`Asn1Real`**`(double value_)`

This constructor creates an REAL object from a double value.

Parameters:

`value_` - double value

(continued from last page)

Asn1Real

```
public Asn1Real(boolean floatType)
```

Deprecated. *use Asn1Real() or Asn1Real(double) instead.*

Parameters:

`floatType` - ignored

Methods

getAsn1TypeName

```
public java.lang.String getAsn1TypeName()
```

Returns the ASN.1 type name.

Returns:

The ASN.1 type name REAL.

---

decode

```
protected final void decode(Asn1DecodeBuffer buffer,
         int length,
         int baseflag)
  throws java.io.IOException
```

This method decodes the content octets of an ASN.1 REAL value into this object, where the REAL was encoded as for BER and the length is taken to be as given. Note that this method is used for OER also, since OER uses the same content octets as BER, at least in certain cases.

Parameters:

`buffer` - Decode message buffer object

`length` - Length of contents

`baseflag` - ANY_BASE: encoding may use any base ANY_BINARY: encoding may any of bases 2, 8, 16 BASE_2_ONLY: encoding may only use base 2

Throws:

`java.io.IOException` - for any I/O error

---

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

(continued from last page)

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of component or negative status value

---

decode

```
protected final void decode(Asn1PerDecodeBuffer buffer,
         int baseflag)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' in this object.

Parameters:

`buffer` - PER Decode message buffer object

`baseflag` - ANY_BASE: encoding may use base 2 or base 10 BASE_2_ONLY: encoding may only use base 2

Throws:

`java.io.IOException` - for any I/O error

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' in this object.

Parameters:

`buffer` - PER Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

(continued from last page)

Decode a REAL value, encoded according to OER, into this object. This method applies to unconstrained REAL values and REAL values that are constrained but not meeting the requirements for encoding using IEEE 754 single or double precision format.

Parameters:

`buffer` - The decode buffer.

Throws:

`java.io.IOException` - for any I/O error

decodeSingle

```
public final void decodeSingle(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a REAL value, encoded according to OER in single precision format.

Parameters:

`buffer` - The decode buffer.

Throws:

`java.io.IOException` - for any I/O error

decodeDouble

```
public final void decodeDouble(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a REAL value, encoded according to OER in double precision format.

Parameters:

`buffer` - The decode buffer.

Throws:

`java.io.IOException` - for any I/O error

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode this REAL value, according to OER, into the buffer. This method applies to unconstrained REAL values and REAL values that are constrained but not meeting the requirements for encoding using IEEE 754 single or double precision format.

Parameters:

`buffer` - The encode buffer.

encodeSingle

```
public final void encodeSingle(Asn1OerEncodeBuffer buffer)
```

Encode this REAL value, according to OER, in single precision format, into the buffer.

Parameters:

`buffer` - The encode buffer.

encodeDouble

```
public final void encodeDouble(Asn1OerEncodeBuffer buffer)
```

Encode this REAL value, according to OER, in double precision format, into the buffer.

(continued from last page)

Parameters:

`buffer` - The encode buffer.

---

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs,
         boolean decodingElemName,
         boolean modifiedEncodings)
  throws Asn1Exception
```

This method decodes an ASN.1 real value using XER.

Parameters:

`buffer` - String containing data to be decoded

`attrs` - Attributes string from element tag

`decodingElemName` - Pass true if you the ASN.1 value being decoded was encoded as an empty element and buffer is the element name. Such an encoding occurs for the special real values under basic-XER, canonical-XER, and extended-XER without GLOBAL-DEFAULTS MODIFIED-ENCODINGS present.

`modifiedEncodings` - Pass true if decoding under extended-XER with GLOBAL-DEFAULTS MODIFIED-ENCODINGS present.

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 real value according to Obj-Sys encoding rules. The value is encoded as text.

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix,
         boolean asText)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 real value according to XER encoding rules. It is for use with extended-XER.

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

`nsPrefix` - Namespace prefix for element.

(continued from last page)

`asText` - If TRUE, encode special values as text. Otherwise, special values are encoded as empty elements.

Throws:

`java.io.IOException` - for any I/O error

---

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

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

Throws:

`java.io.IOException` - for any I/O error

---

encodeValue

```
public void encodeValue(Asn1XmlEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

Throws:

`java.io.IOException` - for any I/O error

---

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 real value using the XML schema encoding rules.

Parameters:

`buffer` - String containing data to be decoded

`attrs` - Attributes string from element tag

---

decodeV72

```
public void decodeV72(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 REAL from JSON. This provides ObjSys-specific behavior that predates ITU-T X.697.

Parameters:

`buffer` - The decode buffer.

Throws:

`java.io.IOException` - for any I/O error

(continued from last page)

encodeV72

```
public void encodeV72(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this ASN.1 real value to JSON. This provides ObjSys-specific behavior that predates ITU-T X.697.

Parameters:

`outstream` - The output stream.

Throws:

`java.io.IOException` - for any I/O error

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 REAL from JSON.

Parameters:

`buffer` - The decode buffer.

Throws:

`java.io.IOException` - for any I/O error

---

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this ASN.1 real value to JSON.

Parameters:

`outstream` - The output stream.

Throws:

`java.io.IOException` - for any I/O error

---

equals

```
public boolean equals(double dvalue)
```

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

Parameters:

`dvalue` - REAL test value

Returns:

true if the values are equal

---

equals

```
public boolean equals(java.lang.Object o)
```

This method compares this REAL value to the given Object for equality.

---

hashCode

```
public int hashCode()
```

Returns the hash code for this REAL object.

toString

```
public java.lang.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

normalizedRealValueToString

```
public static java.lang.String normalizedRealValueToString(double value)
```

This method will return a string representation of the normalized REAL value. The output format is [-]X.XXXXE[-]XXX. The format is the ASN.1 value format for this type. This means it is a "NumericRealValue" as defined in X.680. Additionally, if there is a leading minus sign, there will be no whitespace between it and the first digit of the integer part, making it also an "XMLNumericRealValue".

Parameters:

`value` - value to be normalized and stringified.

Returns:

the string described above

encode

```
public void encode(Asn1BerOutputStream out,
                   boolean explicit)
            throws Asn1Exception,
                   java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out)
            throws Asn1Exception,
                   java.io.IOException
```

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

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1Real10

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1UTF8String
            ┗-com.objsys.asn1j.runtime.Asn1Real10
```

public class **Asn1Real10**
extends Asn1UTF8String

This class represents the ASN.1 REAL BASE 10 built-in type.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 9).

Fields inherited from class com.objsys.asn1j.runtime.Asn1UTF8String
TAG

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Real10() The default constructor sets the real10 value to zero.
public	Asn1Real10(java.lang.String data) This constructor creates an real10 object from a string value.

Method Summary

void	convertToDecimal() This method convert the contained real10 value to XML decimal.

void	`convertToNR3Form(boolean cxerForm)` This method convert the contained real10 value to NR3 form.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 REAL from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 REAL value, having base 10, that was encoded according to OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an real10 value using the Packed Encoding Rules (PER).
void	`decodeV72(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 REAL from JSON.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes and writes to the stream an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1CerOutputStream out,boolean explicit)` This method encodes and writes to the stream an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	`encode(Asn1DerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1JsonOutputStream outstream)` Encode this ASN.1 real value to JSON.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 REAL value, whose base is 10, according to OER.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an real10 value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 real10 value using the Packed Encoding Rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 integer value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard(asn2xsd).

void	encodeAttribute(Asn1XmlEncoder buffer,java.lang.String attrName) This method encodes an ASN.1 integer value using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	encodeV72(Asn1JsonOutputStream outstream) Encode this ASN.1 real value to JSON.
byte	getNumberForm() This method calculates the number form of the contained real10 value.
static byte	getNumberForm(java.lang.String value) This method calculates the number form of an real10 value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1UTF8String

decode, decode, decode, decode, decode, decode, decodeUTF8, decodeUTF8, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, setAnyAttribute

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString

decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 9).

Constructors

Asn1Real10

public **Asn1Real10**()

The default constructor sets the real10 value to zero.

Asn1Real10

```
public Asn1Real10(java.lang.String data)
```

This constructor creates an real10 object from a string value.

Parameters:

`data` - String value

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 real10 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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

This method encodes an ASN.1 real10 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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of component or negative status value

encode

```
public int encode(Asn1DerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

This method encodes an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Distinguished Encoding Rules (DER).

Parameters:

`buffer` - Encode message buffer object

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of component or negative status value

(continued from last page)

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an real10 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 'value' in this object.

Parameters:

`buffer` - PER Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an real10 value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters:

`buffer` - PER Encode message buffer object

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 REAL value, having base 10, that was encoded according to OER.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 REAL value, whose base is 10, according to OER.

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

(continued from last page)

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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` - XML element name space prefix

---

encodeAttribute

```
public void encodeAttribute(Asn1XmlEncoder buffer,
         java.lang.String attrName)
  throws Asn1Exception,
         java.io.IOException
```

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
`attrName` - Attribute name

---

decodeV72

```
public void decodeV72(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 REAL from JSON. This provides ObjSys-specific behavior that predates ITU-T X.697.

Parameters:

`buffer`

Throws:

`java.io.IOException`

---

encodeV72

```
public void encodeV72(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this ASN.1 real value to JSON. This provides ObjSys-specific behavior that predates ITU-T X.697.

Parameters:

`outstream`

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 REAL from JSON.

Parameters:

`buffer`

Throws:

`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this ASN.1 real value to JSON.

Parameters:

`outstream`

getNumberForm

```
public static byte getNumberForm(java.lang.String value)
```

This method calculates the number form of an real10 value.

Parameters:

`value` - Real10 value in which to count bits.

Returns:

Number form.

getNumberForm

```
public byte getNumberForm()
```

This method calculates the number form of the contained real10 value.

Returns:

Number form.

convertToNR3Form

```
public void convertToNR3Form(boolean cxerForm)
```

This method convert the contained real10 value to NR3 form. NR3 form number placed in mStringBuffer field.

convertToDecimal

```
public void convertToDecimal()
```

This method convert the contained real10 value to XML decimal. Result number placed in mStringBuffer field.

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes and writes to the stream an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

encode

```
public void encode(Asn1CerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes and writes to the stream an ASN.1 real10 value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Canonical Encoding Rules (CER).

Parameters:

`out` - CER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 real10 value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters:

`out` - PER Encode message buffer object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1RealBase2

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1Real
         ┗-com.objsys.asn1j.runtime.Asn1RealBase2
```

public class **Asn1RealBase2**
extends Asn1Real

Represents an ASN.1 REAL value that has base 2. This class is used when decoding to allow decoding to report errors if the base in the encoding is not legal for base 2 values.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Real`
`ANY_BASE`, `ANY_BINARY`, `BASE_2_ONLY`, `TAG`, `value`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`BIT_STRING`, `BMPString`, `BOOLEAN`, `DATE`, `DATE_TIME`, `DURATION`, `ENUMERATED`, `EOC`, `EXTERNAL`, `GeneralString`, `GeneralTime`, `GraphicString`, `IA5String`, `INTEGER`, `mNonParameterizedTypeName`, `NULL`, `NumericString`, `OBJECT_IDENTIFIER`, `ObjectDescriptor`, `OCTET_STRING`, `OID_IRI`, `OpenType`, `PrintableString`, `REAL`, `RELATIVE_OID_IRI`, `RelativeOID`, `SEQUENCE`, `SET`, `T61String`, `TeletexString`, `TIME`, `TIME_OF_DAY`, `UniversalString`, `UTCTime`, `UTF8String`, `VideotexString`, `VisibleString`

Constructor Summary

`public`	`Asn1RealBase2()` The default constructor sets the double value to zero.
`public`	`Asn1RealBase2(double value_)` This constructor creates an REAL object from a double value.

Method Summary

`void`	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 REAL value including the UNIVERSAL tag value and length if explicit tagging is specified.
`void`	`decode(Asn1OerDecodeBuffer buffer)` Decode a REAL value, encoded according to OER, into this object.
`void`	`decode(Asn1PerDecodeBuffer buffer)` This method decodes ASN.1 REAL value using the Packed Encoding Rules (PER).

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Real`

decode, decode, decode, decode, decode, decode, decodeDouble, decodeSingle, decodeV72, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAttribute, encodeDouble, encodeSingle, encodeV72, encodeValue, equals, equals, getAsn1TypeName, hashCode, normalizedRealValueToString, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1RealBase2

`public` **`Asn1RealBase2`**`()`

The default constructor sets the double value to zero.

Asn1RealBase2

`public` **`Asn1RealBase2`**`(double value_)`

This constructor creates an REAL object from a double value.

Parameters:

`value_` - double value

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

(continued from last page)

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode a REAL value, encoded according to OER, into this object. This method applies to REAL values constrained to base 2 but not meeting the requirements for encoding using IEEE 754 single or double precision format.

Parameters:

`buffer`

Throws:

`IOException`

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' in this object. This method requires the encoding to be base 2, which must be the case when the type is constrained to base 2.

Parameters:

`buffer` - PER Decode message buffer object

com.objsys.asn1j.runtime
Class Asn1RELATIVE_OID_IRI

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1OID_IRI
            ┗-com.objsys.asn1j.runtime.Asn1RELATIVE_OID_IRI
```

public class **Asn1RELATIVE_OID_IRI**
extends Asn1OID_IRI

This class represents the ASN.1 RELATIVE-OID-IRI. It is really nothing more than a string, but it provides the decoding and encoding methods for various encoding rules. That this class extends Asn1OID_IRI should be considered an implementation detail. The reason for the inheritance is that the encodings are the same, except for the BER tag used.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 36).

Fields inherited from class com.objsys.asn1j.runtime.Asn1OID_IRI

TAG

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1RELATIVE_OID_IRI() The default constructor creates an empty string object.
public	Asn1RELATIVE_OID_IRI(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

Asn1Tag	getTag() Allows subclass to use a different tag by overriding this method.
short	getTagNumber() Allows subclass to use a different tag number by overriding this method.

Methods inherited from class com.objsys.asn1j.runtime.Asn1OID_IRI
decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, getTag, getTagNumber

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 36).

Constructors

Asn1RELATIVE_OID_IRI

public **Asn1RELATIVE_OID_IRI**()

The default constructor creates an empty string object.

Asn1RELATIVE_OID_IRI

public **Asn1RELATIVE_OID_IRI**(java.lang.String data)

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

Methods

(continued from last page)

getTagNumber

`protected short` **`getTagNumber`**`()`

Allows subclass to use a different tag number by overriding this method.

Returns:

---

getTag

`protected Asn1Tag` **`getTag`**`()`

Allows subclass to use a different tag by overriding this method.

Returns:

com.objsys.asn1j.runtime
Class Asn1RelativeOID

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1ObjectIdentifier
         ┗-com.objsys.asn1j.runtime.Asn1RelativeOID
```

public class **Asn1RelativeOID**
extends Asn1ObjectIdentifier

This is a container class for holding the components of an ASN.1 relative object identifier value.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 13).

Fields inherited from class com.objsys.asn1j.runtime.Asn1ObjectIdentifier

MAXSUBIDS, TAG, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1RelativeOID() This constructor creates an empty object identifier that can be used in a decode method call to receive an OID value.
public	Asn1RelativeOID(int[] value_) This constructor initializes the object identifier from the given array of integer subidentifier values.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 relative object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.

void	`decode(Asn1OerDecodeBuffer buffer)` Decode an ASN.1 RELATIVE-OID that was encoded according to the Octet Encoding Rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 relative object identifier value using the packed encoding rules (PER).
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 RELATIVE-OID value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 RELATIVE-OID value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 relative object identifier value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 RELATIVE-OID according to Octet Encoding Rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 relative object identifier value using the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 relative object identifier value using the packed encoding rules (PER).
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 RELATIVE-OID value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 RELATIVE-OID value using the XML Encoding as specified in the XML schema standard(asn2xsd).
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
void	`validate()`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1ObjectIdentifier`

`append, decode, decode, decode, decode, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, getAsn1TypeName, hashCode, toString, toXMLValue, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 13).

Constructors

Asn1RelativeOID

```
public Asn1RelativeOID()
```

This constructor creates an empty object identifier that can be used in a decode method call to receive an OID value.

Asn1RelativeOID

```
public Asn1RelativeOID(int[] value_)
```

This constructor initializes the object identifier from the given array of integer subidentifier values.

Parameters:

`value_` - Array of subidentifiers

Methods

getAsn1TypeName

```
public java.lang.String getAsn1TypeName()
```

Returns the ASN.1 type name.

Returns:

The ASN.1 type name RELATIVE-OID.

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length of encoded component in octets

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 relative object identifier value using the packed encoding rules (PER).

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`public member variable within this class.

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode an ASN.1 RELATIVE-OID that was encoded according to the Octet Encoding Rules (OER).

Parameters:

`buffer`

Throws:

`java.io.IOException`

(continued from last page)

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 RELATIVE-OID according to Octet Encoding Rules (OER).

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`elemName` - Element name

---

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

---

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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 prefix value

---

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 RELATIVE-OID value using the XML schema encoding rules.

Parameters:

`buffer` - String containing data to be decoded

`attrs` - Attributes string from element tag

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`public member variable within this class.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

validate

```
protected void validate()
```

com.objsys.asn1j.runtime
Class Asn1Seq

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1SeqOrSet
         ┗-com.objsys.asn1j.runtime.Asn1Seq
```

public abstract class **Asn1Seq**
extends Asn1SeqOrSet

Asn1Seq is a base class for classes representing ASN.1 sequences. It provides some abstract methods by which elements can be accessed generically.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Seq()

Methods inherited from class com.objsys.asn1j.runtime.Asn1SeqOrSet
getElementCount, getElementName, getElementValue, getElementValue

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

Asn1Seq

```
public Asn1Seq()
```

com.objsys.asn1j.runtime
Class Asn1SeqOf

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Type
      |
      +-com.objsys.asn1j.runtime.Asn1SeqSetOf
         |
         +-com.objsys.asn1j.runtime.Asn1SeqOf
```

public abstract class **Asn1SeqOf**
extends Asn1SeqSetOf

Asn1SeqOf is a base class for classes representing ASN.1 sequence of types It provides an abstract methods by which elements can be accessed generically.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1SeqOf()

Methods inherited from class com.objsys.asn1j.runtime.Asn1SeqSetOf
getElementValues

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

Asn1SeqOf

```
public Asn1SeqOf()
```

com.objsys.asn1j.runtime
Class Asn1SeqOrderException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1SeqOrderException
```

public class **Asn1SeqOrderException**
extends Asn1Exception

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..

Constructor Summary

public	`Asn1SeqOrderException()` This constructor creates an exception object with a textual message describing the element that was received out-of-order.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1SeqOrderException

`public` **`Asn1SeqOrderException`**`()`

This constructor creates an exception object with a textual message describing the element that was received out-of-order.

com.objsys.asn1j.runtime
Class Asn1SeqOrSet

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Type
      |
      +-com.objsys.asn1j.runtime.Asn1SeqOrSet
```

Direct Known Subclasses:
Asn1Set, Asn1Seq

---

public abstract class **Asn1SeqOrSet**
extends Asn1Type

Asn1SeqOrSet is a base class for classes representing ASN.1 sequences and sets. It provides some abstract methods by which elements can be accessed generically.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1SeqOrSet()

Method Summary

abstract int	getElementCount() Return the number of declared elements, plus 1 if the sequence/set is extensible.
abstract java.lang.String	getElementName(int index) Return the ASN.1 name of the element identified by index.
abstract java.lang.Object	getElementValue(int index) Return the value of the element identified by index.
java.lang.Object	getElementValue(java.lang.String name) Return the value of the element identified by the given name.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1SeqOrSet

`public` **`Asn1SeqOrSet`**`()`

Methods

getElementCount

`public abstract int` **`getElementCount`**`()`

Return the number of declared elements, plus 1 if the sequence/set is extensible.

getElementValue

`public abstract java.lang.Object` **`getElementValue`**`(int index)`

Return the value of the element identified by index. Each declared element in the sequence/set is assigned an index, beginning with zero, following the textual order of the declarations. If the element is optional and not present, null is returned.

Returns:

Object The value of the element or else null. The return type is Object rather than Asn1Type because sometimes the object to be returned is an array. The returned object will be the declared type of the corresponding element in the generated code.

getElementName

`public abstract java.lang.String` **`getElementName`**`(int index)`

Return the ASN.1 name of the element identified by index. Each declared element in the sequence/set is assigned an index, beginning with zero, following the textual order of the declarations. If the element name is unknown (as for undeclared extension elements), null is returned.

getElementValue

`public java.lang.Object` **`getElementValue`**`(java.lang.String name)`

Return the value of the element identified by the given name. If the the sequence or set does not contain an element declaration with the given name, or if such element is optional and not present, null is returned.

Parameters:

`name`

Returns:

Asn1Type the value of the element or else null

com.objsys.asn1j.runtime
Class Asn1SeqSetOf

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Type
      |
      +-com.objsys.asn1j.runtime.Asn1SeqSetOf
```

Direct Known Subclasses:
Asn1SetOf, Asn1SeqOf

---

public abstract class **Asn1SeqSetOf**
extends Asn1Type

Asn1SeqSetOf is a base class for classes representing ASN.1 sequence of and set of types. It provides an abstract methods by which elements can be accessed generically.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1SeqSetOf()

Method Summary

abstract Asn1Type[]	getElementValues() Return the array of elements.

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

---

Constructors

(continued from last page)

Asn1SeqSetOf

```
public Asn1SeqSetOf()
```

Methods

getElementValues

```
public abstract Asn1Type[] getElementValues()
```

Return the array of elements.

com.objsys.asn1j.runtime
Class Asn1Set

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1SeqOrSet
         ┗-com.objsys.asn1j.runtime.Asn1Set
```

public abstract class **Asn1Set**
extends Asn1SeqOrSet

Asn1Set is a base class for classes representing ASN.1 sets. It provides some abstract methods by which elements can be accessed generically.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Set()

Methods inherited from class com.objsys.asn1j.runtime.Asn1SeqOrSet
getElementCount, getElementName, getElementValue, getElementValue

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

Asn1Set

```
public Asn1Set()
```

com.objsys.asn1j.runtime
Class Asn1SetDuplicateException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1SetDuplicateException
```

public class **Asn1SetDuplicateException**
extends Asn1Exception

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

Constructor Summary

public	`Asn1SetDuplicateException(Asn1BerDecodeBuffer buffer,Asn1Tag tag)` This constructor creates an exception object with a textual message describing the tag of the duplicate element..
public	`Asn1SetDuplicateException(java.lang.String message)`

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1SetDuplicateException

```
public Asn1SetDuplicateException(Asn1BerDecodeBuffer buffer,
                                 Asn1Tag tag)
```

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

Parameters:

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

Asn1SetDuplicateException

```
public Asn1SetDuplicateException(java.lang.String message)
```

com.objsys.asn1j.runtime
Class Asn1SetOf

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1SeqSetOf
         ┗-com.objsys.asn1j.runtime.Asn1SetOf
```

public abstract class **Asn1SetOf**
extends Asn1SeqSetOf

Asn1SetOf is a base class for classes representing ASN.1 set of types. It provides an abstract methods by which elements can be accessed generically.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1SetOf()

Methods inherited from class com.objsys.asn1j.runtime.Asn1SeqSetOf
getElementValues

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

Asn1SetOf

```
public Asn1SetOf()
```

com.objsys.asn1j.runtime
Class Asn1SizeConstraint

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1SizeConstraint
```

public class **Asn1SizeConstraint**
extends java.lang.Object

This class is used to represent a size constraint. Only sizes represented as a single value or range are supported.

This class is mainly for internal use by the compiler when generating methods that encode/decode PER SEQUENCE OF components containing size constraints.

Field Summary

boolean	mExtensible
long	mExtLower
long	mExtUpper
long	mRootLower
long	mRootUpper

Constructor Summary

public	Asn1SizeConstraint(long lower,long upper) This constructor sets the range values for a non-extensisble constraint.
public	Asn1SizeConstraint(long lower,long upper,long extLower,long extUpper) This constructor sets the range values for an extensisble constraint.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mRootLower

protected long **mRootLower**

mRootUpper

```
protected long mRootUpper
```

mExtLower

```
protected long mExtLower
```

mExtUpper

```
protected long mExtUpper
```

mExtensible

```
protected boolean mExtensible
```

Constructors

Asn1SizeConstraint

```
public Asn1SizeConstraint(long lower,
                          long upper)
```

This constructor sets the range values for a non-extensisble constraint.

Parameters:

`lower` - Range lower root value
`upper` - Range upper root value

Asn1SizeConstraint

```
public Asn1SizeConstraint(long lower,
                          long upper,
                          long extLower,
                          long extUpper)
```

This constructor sets the range values for an extensisble constraint.

Parameters:

`lower` - Range lower root value
`upper` - Range upper root value
`extLower` - Range lower extend value
`extUpper` - Range upper extended value

com.objsys.asn1j.runtime
Class Asn1Status

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Status
```

public class **Asn1Status**
extends java.lang.Object

This class defines common constants used in the run-time and generated code. Note that all error reporting in the Java version is done via exceptions. Therefore, there ar very few status values defined in the class.

Field Summary

static final int	INDEFLEN This constant indicates an indefinite length field was parsed Value: **-9999**

Constructor Summary

public	Asn1Status()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

INDEFLEN

public static final int **INDEFLEN**

This constant indicates an indefinite length field was parsed

Constructors

Asn1Status

public **Asn1Status**()

com.objsys.asn1j.runtime
Class Asn1T61String

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1VarWidthCharString
            ┗-com.objsys.asn1j.runtime.Asn1T61String
```

public class **Asn1T61String**
extends Asn1VarWidthCharString

This is a container class for holding the components of an ASN.1 teletex (T61) string value.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 20).

Fields inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1T61String() The default constructor creates an empty string object.
public	Asn1T61String(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 T61String from JSON.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes and writes to the stream an ASN.1 T61 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1JsonOutputStream outstream)` Encode this T61String to JSON.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1VarWidthCharString`

`decode, decode, decode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 20).

Constructors

(continued from last page)

Asn1T61String

`public` **`Asn1T61String`**`()`

The default constructor creates an empty string object.

Asn1T61String

`public` **`Asn1T61String`**`(java.lang.String data)`

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes and writes to the stream an ASN.1 T61 string value including the UNIVERSAL tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 T61String from JSON.

Parameters:

`buffer` - The buffer to decode from.

Throws:

`java.io.IOException` - for I/O exception

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this T61String to JSON.

Parameters:

`outstream` - The stream to encode to.

Throws:

`java.io.IOException` - for I/O exception

com.objsys.asn1j.runtime
Class Asn1Tag

```
java.lang.Object
    |
    +-com.objsys.asn1j.runtime.Asn1Tag
```

All Implemented Interfaces:
java.io.Serializable

---

public class **Asn1Tag**
extends java.lang.Object
implements java.io.Serializable

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

Field Summary

static final short	APPL Mask value for an APPLICATION tag Value: **64**
static final short	Bit8Mask Bit 8 (MSB) octet mask value Value: **128**
static final short	ClassMask Mask value to mask the class bits from a tag Value: **192**
static final short	CONS Mask value for CONSTRUCTED form Value: **32**
static final short	CTXT Mask value for a context-specific tag Value: **128**
static final Asn1Tag	ENUM ASN.1 ENUMERATED type tag value
static final Asn1Tag	EOC ASN.1 end-of-contents (EOC) type tag value
static final boolean	EXPL This specifies that explicit tagging should be used. Value: **true**

static final short	EXTIDCODE Mask value for extended tag identifier indicator Value: **31**
static final short	FormMask Mask value to mask the form bit from a tag Value: **32**
static final short	IDMask Mask value to mask the tag ID bits from a tag Value: **31**
static final boolean	IMPL This specifies that implicit tagging should be used. Value: **false**
static final short	L7BitsMask Lower 7 bits octet mask value Value: **127**
short	mClass Tag class value (UNIV, APPL, CTXT, or PRIV)
short	mForm Tag form value (PRIM or CONS)
int	mIDCode Tag ID code
static final short	PRIM Mask value for PRIMITIVE form Value: **0**
static final short	PRIV Mask value for a PRIVATE tag Value: **192**
static final Asn1Tag	SEQUENCE ASN.1 SEQUENCE type tag value
static final Asn1Tag	SET ASN.1 SET type tag value
static final short	UNIV Mask value for a UNIVERSAL tag Value: **0**

Constructor Summary

public	Asn1Tag() The default constructor initializes all fields to zero.

public	`Asn1Tag(Asn1Tag _tag)` The copy constructor initializes all fields to those from the given tag parameter.
public	`Asn1Tag(short class_,short form,int idCode)` This constructor initializes all fields to the given values

Method Summary

boolean	`equals(java.lang.Object _tag)` This method compares this tag with the given tag value for equality.
boolean	`equals(short class_,short form,int idCode)` This method compares this tag with the given tag value for equality.
boolean	`isConstructed()` This method tests if the tag is constructed.
boolean	`isEOC()` This method tests if the tag is an end-of-contents (EOC) tag.
java.lang.String	`toString()` This method will return a formatted string representing the tag value.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

EXPL

`public static final boolean` **`EXPL`**

This specifies that explicit tagging should be used.

IMPL

`public static final boolean` **`IMPL`**

This specifies that implicit tagging should be used.

UNIV

`public static final short` **`UNIV`**

Mask value for a UNIVERSAL tag

APPL

`public static final short` **`APPL`**

Mask value for an APPLICATION tag

CTXT

`public static final short CTXT`

Mask value for a context-specific tag

PRIV

`public static final short PRIV`

Mask value for a PRIVATE tag

ClassMask

`public static final short ClassMask`

Mask value to mask the class bits from a tag

PRIM

`public static final short PRIM`

Mask value for PRIMITIVE form

CONS

`public static final short CONS`

Mask value for CONSTRUCTED form

FormMask

`public static final short FormMask`

Mask value to mask the form bit from a tag

EXTIDCODE

`public static final short EXTIDCODE`

Mask value for extended tag identifier indicator

IDMask

`public static final short IDMask`

Mask value to mask the tag ID bits from a tag

ENUM

`public static final com.objsys.asn1j.runtime.Asn1Tag ENUM`

ASN.1 ENUMERATED type tag value

EOC

`public static final com.objsys.asn1j.runtime.Asn1Tag EOC`

ASN.1 end-of-contents (EOC) type tag value

(continued from last page)

SEQUENCE

```
public static final com.objsys.asn1j.runtime.Asn1Tag SEQUENCE
```

ASN.1 SEQUENCE type tag value

SET

```
public static final com.objsys.asn1j.runtime.Asn1Tag SET
```

ASN.1 SET type tag value

Bit8Mask

```
public static final short Bit8Mask
```

Bit 8 (MSB) octet mask value

L7BitsMask

```
public static final short L7BitsMask
```

Lower 7 bits octet mask value

mClass

```
public transient short mClass
```

Tag class value (UNIV, APPL, CTXT, or PRIV)

mForm

```
public transient short mForm
```

Tag form value (PRIM or CONS)

mIDCode

```
public transient int mIDCode
```

Tag ID code

Constructors

Asn1Tag

```
public Asn1Tag()
```

The default constructor initializes all fields to zero.

Asn1Tag

```
public Asn1Tag(Asn1Tag _tag)
```

The copy constructor initializes all fields to those from the given tag parameter.

Parameters:

`_tag` - The input tag value to copy.

(continued from last page)

Asn1Tag

```
public Asn1Tag(short class_,
               short form,
               int idCode)
```

This constructor initializes all fields to the given values

Parameters:

`class_` - Tag class value (UNIV, APPL, CTXT, or PRIV)
`form` - Tag form value (PRIM or CONS)
`idCode` - Tag identifier code

Methods

equals

```
public boolean equals(short class_,
         short form,
         int idCode)
```

This method compares this tag with the given tag value for equality.

Parameters:

`class_` - Tag class value (UNIV, APPL, CTXT, or PRIV)
`form` - Tag form value (PRIM or CONS)
`idCode` - Tag identifier code

Returns:

True if tags are equal

---

equals

```
public boolean equals(java.lang.Object _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 tags are equal

---

isConstructed

```
public boolean isConstructed()
```

This method tests if the tag is constructed.

Returns:

True if tag is constructed.

---

isEOC

```
public boolean isEOC()
```

This method tests if the tag is an end-of-contents (EOC) tag.

Returns:

True if tag is an EOC.

toString

`public java.lang.String` **`toString`**`()`

This method will return a formatted string representing the tag value. The form is "[<class> <ID>]" (i.e. the ASN.1 standard syntax for a tag value).

Returns:

Formatted tag string

com.objsys.asn1j.runtime
Interface Asn1TaggedEventHandler

All Known Implementing Classes:
Asn1BerMessageDumpHandler

---

public interface **Asn1TaggedEventHandler**

This interface defines the methods that must be implemented to define a SAX-like event handler. These methods are invoked from within the generated Java decode logic when significant events occur during the parsing of an ASN.1 message.

A tagged event handler differs from a named event handler in that it returns the tags from within a BER or DER message instead of the symbolic names. This type of handler can be used to generically parse a message without knowledge of the associated ASN.1 schema definition. It is used in conjunction with the Asn1BerDecodeBuffer *parse*method.

Method Summary

abstract void	`contents(byte[] data)` The contents callback method is invoked when the contents of a primitive data element are parsed.
abstract void	`endElement(Asn1Tag tag)` The endElement callback method is invoked when the end of a tagged element is parsed.
abstract void	`startElement(Asn1Tag tag,int len,byte[] tagLenBytes)` The startElement callback method is invoked when the start of any tagged element is parsed.

Methods

startElement

```
public abstract void startElement(Asn1Tag tag,
          int len,
          byte[] tagLenBytes)
```

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

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

---

endElement

```
public abstract void endElement(Asn1Tag tag)
```

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

Parameters:
`tag` - Parsed tag value.

contents

```
public abstract void contents(byte[] data)
```

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

Parameters:

`data` - Array containing encoded contents bytes.

com.objsys.asn1j.runtime
Class Asn1TagMatchFailedException

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1TagMatchFailedException
```

public class **Asn1TagMatchFailedException**
extends Asn1Exception

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

Constructor Summary

public	`Asn1TagMatchFailedException(Asn1BerDecodeBuffer buffer,Asn1Tag expectedTag,Asn1Tag parsedTag)` This constructor creates an exception object with a textual message describing the expected and parsed tag values..
public	`Asn1TagMatchFailedException(Asn1BerDecodeBuffer buffer,Asn1Tag expectedTag)` This constructor creates an exception object with a textual message describing only the expected tag value.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1TagMatchFailedException

```
public Asn1TagMatchFailedException(Asn1BerDecodeBuffer buffer,
                                   Asn1Tag expectedTag,
                                   Asn1Tag parsedTag)
```

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

(continued from last page)

Parameters:

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

Asn1TagMatchFailedException

```
public Asn1TagMatchFailedException(Asn1BerDecodeBuffer buffer,
                                   Asn1Tag expectedTag)
```

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

Parameters:

`buffer` - BER decode buffer object reference
`expectedTag` - Expected tag value

com.objsys.asn1j.runtime
Class Asn1TBCDString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
         ┗-com.objsys.asn1j.runtime.Asn1TBCDString
```

public class **Asn1TBCDString**
extends Asn1OctetString

Asn1TBCDString represents an OCTET STRING with an overridden toString method that converts the bytes to a character string of [0-9*#abc] characters.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OctetString`
`TAG`, `value`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`BIT_STRING`, `BMPString`, `BOOLEAN`, `DATE`, `DATE_TIME`, `DURATION`, `ENUMERATED`, `EOC`, `EXTERNAL`, `GeneralString`, `GeneralTime`, `GraphicString`, `IA5String`, `INTEGER`, `mNonParameterizedTypeName`, `NULL`, `NumericString`, `OBJECT_IDENTIFIER`, `ObjectDescriptor`, `OCTET_STRING`, `OID_IRI`, `OpenType`, `PrintableString`, `REAL`, `RELATIVE_OID_IRI`, `RelativeOID`, `SEQUENCE`, `SET`, `T61String`, `TeletexString`, `TIME`, `TIME_OF_DAY`, `UniversalString`, `UTCTime`, `UTF8String`, `VideotexString`, `VisibleString`

Constructor Summary

public	`Asn1TBCDString()` This constructor creates an empty octet string that can be used in a decode method call to receive an octet string value.
public	`Asn1TBCDString(byte[] data)` This constructor initializes an octet string from the given byte array.
public	`Asn1TBCDString(byte[] data,int offset,int nbytes)` This constructor initializes an octet string from a portion of the given byte array.
public	`Asn1TBCDString(java.lang.String value_)` This constructor parses the given TBCD character string and assigns the value to the internal octet string.

Method Summary

`java.lang.String`	`toString()` This method will return a string representation of the octet string, using Asn1Util.TBCDToString.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1OctetString`

compareTo, decode, decode, decode, decode, decode, decode, decode, decode, decodeAsBase64, decodeAsHex, decodeContent, decodeRemainingBits, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsBase64, encodeAsHex, encodeAttribute, encodeBase64Binary, encodeContent, equals, equals, equals, getAsn1TypeName, getLength, getMderLength, hashCode, toInputStream, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1TBCDString

```
public Asn1TBCDString()
```

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

Asn1TBCDString

```
public Asn1TBCDString(byte[] data)
```

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

Parameters:

`data` - Byte array containing an octet string in binary form.

Asn1TBCDString

```
public Asn1TBCDString(byte[] data,
                      int offset,
                      int nbytes)
```

This constructor initializes an octet string from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes.

Parameters:

`data` - Byte array containing an octet string in binary form.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array

Asn1TBCDString

```
public Asn1TBCDString(java.lang.String value_)
```

This constructor parses the given TBCD character string and assigns the value to the internal octet string.

Parameters:

(continued from last page)

`value_` - TBCD character string. E.g. "5551212*c"

Methods

toString

`public java.lang.String` **`toString`**`()`

This method will return a string representation of the octet string, using Asn1Util.TBCDToString. This follows the encoding of TBCD specified in 3GPP 29.002.

com.objsys.asn1j.runtime
Class Asn1Time

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1Time
```

public class **Asn1Time**
extends Asn18BitCharString

This class is used for all TIME types that are not one of the useful time types (viz. DATE, DATE-TIME, TIME-OF-DAY, DURATION).

Field Summary

static final int	Apr Value: **4**
static final int	April Value: **4**
static final int	Aug Value: **8**
static final int	August Value: **8**
static final int	Dec Value: **12**
static final int	December Value: **12**
static final int	Feb Value: **2**
static final int	February Value: **2**
static final int	Jan Value: **1**

static final int	January Month constants. Value: **1**
static final int	Jul Value: **7**
static final int	July Value: **7**
static final int	Jun Value: **6**
static final int	June Value: **6**
static final int	Mar Value: **3**
static final int	March Value: **3**
static final int	May Value: **5**
static final int	Nov Value: **11**
static final int	November Value: **11**
static final int	Oct Value: **10**
static final int	October Value: **10**
static final int	Sep Value: **9**
static final int	September Value: **9**
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 14).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1Time() The default constructor creates an empty string object.
public	Asn1Time(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an ASN.1 TIME string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	encode(Asn1BerEncodeBuffer buffer,boolean explicit) This method encodes an ASN.1 TIME string type.
void	encode(Asn1BerOutputStream out,boolean explicit) This method encodes and writes to the stream an ASN.1 DATE value including the UNIVERSAL tag value and length if explicit tagging is specified.

Methods inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

January

`public static final int` **`January`**

Month constants. These were defined in what used to be Asn1Time and was renamed to Asn1AbstractTime. They are here to allow backward compatibility so users don't have to modify their code to use Asn1AbstractTime to refer to these constants.

Jan

`public static final int` **`Jan`**

February

`public static final int` **`February`**

Feb

`public static final int` **`Feb`**

March

`public static final int` **`March`**

Mar

`public static final int` **`Mar`**

April

`public static final int` **`April`**

Apr

`public static final int` **`Apr`**

(continued from last page)

May

```
public static final int May
```

June

```
public static final int June
```

Jun

```
public static final int Jun
```

July

```
public static final int July
```

Jul

```
public static final int Jul
```

August

```
public static final int August
```

Aug

```
public static final int Aug
```

September

```
public static final int September
```

Sep

```
public static final int Sep
```

October

```
public static final int October
```

Oct

```
public static final int Oct
```

November

```
public static final int November
```

Nov

```
public static final int Nov
```

December

```
public static final int December
```

Dec

```
public static final int Dec
```

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 14).

Constructors

Asn1Time

```
public Asn1Time()
```

The default constructor creates an empty string object.

Asn1Time

```
public Asn1Time(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
            throws Asn1Exception,
                   java.io.IOException
```

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

Parameters:

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

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
          boolean explicit)
  throws Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

encode

```
public void encode(Asn1BerOutputStream out,
          boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`java.io.IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1TimeOfDay

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1TimeOfDay
```

public class **Asn1TimeOfDay**
extends Asn18BitCharString

This is a container class for holding the components of an ASN.1 TIME-OF-DAY value.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 32).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString

BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1TimeOfDay() The default constructor creates an empty string object.
public	Asn1TimeOfDay(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 TIME-OF-DAY string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 TIME-OF-DAY value in accordance with the octet encoding rules (OER).
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 TIME-OF-DAY value in accordance with the packed encoding rules (PER).
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 TIME-OF-DAY string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes and writes to the stream an ASN.1 TIME-OF-DAY value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes an ASN.1 TIME-OF-DAY value in accordance with the octet encoding rules (OER).
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 TIME-OF-DAY value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 TIME-OF-DAY value in accordance with the packed encoding rules (PER) directly into the stream.

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`

`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

```
public static final com.objsys.asn1j.runtime.Asn1Tag TAG
```

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 32).

Constructors

Asn1TimeOfDay

```
public Asn1TimeOfDay()
```

The default constructor creates an empty string object.

Asn1TimeOfDay

```
public Asn1TimeOfDay(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
                   java.io.IOException
```

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

Parameters:

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

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
                  boolean explicit)
  throws Asn1Exception
```

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

Parameters:

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

Returns:

Length in octets of encoded component

(continued from last page)

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`java.io.IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 TIME-OF-DAY value in accordance with the octet encoding rules (OER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes an ASN.1 TIME-OF-DAY value in accordance with the octet encoding rules (OER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 TIME-OF-DAY value in accordance with the packed encoding rules (PER). The decoded result is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 TIME-OF-DAY value in accordance with the packed encoding rules (PER). The value to encode is stored in the public `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an ASN.1 TIME-OF-DAY 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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object

Throws:

`java.io.IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1TimeUtil

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1TimeUtil
```

Direct Known Subclasses:
Asn1PerTime

---

public class **Asn1TimeUtil**
extends java.lang.Object

Provides methods to encode and decode TIME values in various encoding rules. The application representation of the TIME value is simply a String, whose content shall be a TimeValue, as defined in X.680. Some notes on formatting: + A year/century should have a leading plus sign if and only if the year/century > 9999/99. + A year/century having 5/3 or more digits should never include leading zeros. A value with a year of 1985 has the property Year="Basic" and is formatted with only 4 digits, never more. + A year/century should never have fewer than four/two digits. This is the only case where leading zeros may be used. + When FRACTION is used to specify a number of fraction digits, the string representation must have precisely that many digits for the fraction. Note that FRACTION is not to be used with DURATION.

Field Summary

static final int	ANY For PER, when ANY is combined with CENTURY/YEAR, it indicates that the type is constrained to values with Year=Ln or NEGATIVE (as for the defined types named ANY-*), while absence of ANY indicates the type is constrained to values with Year=Basic or Proleptic. Value: **32768**
static final int	CENTURY Value: **16384**
static final int	DATE Value: **13312**
static final int	DATE_TIME Value: **14208**
static final int	DAY Value: **1024**
static final int	DIFF Indicates a time includes a timezone difference from UTC. Value: **32**
static final int	DT_ANY_CENTURY Value: **49152**
static final int	DT_ANY_YEAR Value: **40960**

static final int	DT_ANY_YEAR_DAY Value: **41984**
static final int	DT_ANY_YEAR_MONTH Value: **45056**
static final int	DT_ANY_YEAR_MONTH_DAY Value: **46080**
static final int	DT_ANY_YEAR_WEEK Value: **43008**
static final int	DT_ANY_YEAR_WEEK_DAY Value: **44032**
static final int	DT_CENTURY Value: **16384**
static final int	DT_HOURS Value: **512**
static final int	DT_HOURS_AND_DIFF Value: **544**
static final int	DT_HOURS_AND_DIFF_AND_FRACTION Value: **547**
static final int	DT_HOURS_AND_FRACTION Value: **515**
static final int	DT_HOURS_UTC Value: **576**
static final int	DT_HOURS_UTC_AND_FRACTION Value: **579**
static final int	DT_MINUTES Value: **768**
static final int	DT_MINUTES_AND_DIFF Value: **800**
static final int	DT_MINUTES_AND_DIFF_AND_FRACTION Value: **803**
static final int	DT_MINUTES_AND_FRACTION Value: **771**

static final int	DT_MINUTES_UTC Value: **832**
static final int	DT_MINUTES_UTC_AND_FRACTION Value: **835**
static final int	DT_SECONDS Value: **896**
static final int	DT_SECONDS_AND_DIFF Value: **928**
static final int	DT_SECONDS_AND_DIFF_AND_FRACTION Value: **931**
static final int	DT_SECONDS_AND_FRACTION Value: **899**
static final int	DT_SECONDS_UTC Value: **960**
static final int	DT_SECONDS_UTC_AND_FRACTION Value: **963**
static final int	DT_YEAR Value: **8192**
static final int	DT_YEAR_DAY Value: **9216**
static final int	DT_YEAR_MONTH Value: **12288**
static final int	DT_YEAR_MONTH_DAY Value: **13312**
static final int	DT_YEAR_WEEK Value: **10240**
static final int	DT_YEAR_WEEK_DAY Value: **11264**
static final int	DURATION Indicates a duration. Value: **16**

static final int	FRACTION This is a mask that is used to mark off the bits used for the number of decimal places for a fraction. Value: **15**
static final int	HOURS Value: **512**
static final int	MINUTES Value: **256**
static final int	MONTH Value: **4096**
static final int	SECONDS Value: **128**
static final int	TIME_OF_DAY Value: **896**
static final int	UTC Indicates a time is UTC. Value: **64**
static final int	WEEK Week is present. Value: **2048**
static final int	YEAR Value: **8192**

Constructor Summary

public	Asn1TimeUtil()

Method Summary

static java.lang.String	convertToDER(java.lang.String timeValue) Return the DER representation of a TimeValue.
static java.lang.String	decodeDate(Asn1OerDecodeBuffer source,int flags) Decode a date value according to OER from the given source.
static java.lang.String	decodeDate(Asn1PerDecodeBuffer source,int flags) Decode a date value from the given source.
static java.lang.String	decodeDateTime(Asn1OerDecodeBuffer source,int flags) Decode a date-time value from the given source.

static java.lang.String	decodeDateTime(Asn1PerDecodeBuffer source,int flags) Decode a date-time value from the given source.
static java.lang.String	decodeDuration(Asn1OerDecodeBuffer buffer) Decode a duration according to OER from the given buffer.
static java.lang.String	decodeDuration(Asn1PerDecodeBuffer source,boolean recurring) Decode a (possibly recurring) duration from the given source.
static java.lang.String	decodeIntervalDE(Asn1PerDecodeBuffer source,boolean recurring,int flags)
static java.lang.String	decodeIntervalSD(Asn1PerDecodeBuffer source,boolean recurring,int flags)
static java.lang.String	decodeIntervalSE(Asn1PerDecodeBuffer source,boolean recurring,int flags)
static java.lang.String	decodeTime(Asn1OerDecodeBuffer buffer,int flags) Decode a time value according to OER from the given source.
static java.lang.String	decodeTime(Asn1PerDecodeBuffer source,int flags) Decode a time value from the given source.
static void	encodeDate(Asn1OerEncodeBuffer encoder,java.lang.String value,int flags) Encode a date value according to OER, using the given encoder.
static void	encodeDate(Asn1PerEncoder encoder,java.lang.String value,int flags) Encode a date value according to PER, using the given encoder.
static void	encodeDateTime(Asn1OerEncodeBuffer buffer,java.lang.String value,int flags) Encode a datetime value according to OER.
static void	encodeDateTime(Asn1PerEncoder encoder,java.lang.String value,int flags) Encode a datetime value.
static void	encodeDuration(Asn1OerEncodeBuffer buffer,java.lang.String value) Encode a duration according to OER.
static void	encodeDuration(Asn1PerEncoder encoder,java.lang.String value,boolean recurring) Encode a duration according to PER.
static void	encodeIntervalDE(Asn1PerEncoder encoder,java.lang.String value,boolean recurring,int flags)
static void	encodeIntervalSD(Asn1PerEncoder encoder,java.lang.String value,boolean recurring,int flags)
static void	encodeIntervalSE(Asn1PerEncoder encoder,java.lang.String value,boolean recurring,int flags)

static void	encodeTime(Asn1OerEncodeBuffer buffer,java.lang.String value,int flags) Encode a time value according to OER.
static void	encodeTime(Asn1PerEncoder encoder,java.lang.String value,int flags) Encode a time value.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

ANY

public static final int **ANY**

For PER, when ANY is combined with CENTURY/YEAR, it indicates that the type is constrained to values with Year=Ln or NEGATIVE (as for the defined types named ANY-*), while absence of ANY indicates the type is constrained to values with Year=Basic or Proleptic. There is a little inconsistency here, however. At present, code generation will use the DT_ANY* constants for a type that has Year=Negative, or that has Year=Ln for some n. By contrast, the defined types named ANY-* include values that have either Year=Negative or Year=L5. I think this discrepancy doesn't matter though, because PER uses the same encodings whether type includes: - only values with Year=Ln - only values with Year=L5 - only values with Year=Negative - only values with Year=Negative or Year=L5 The ANY flag's presence/absence is irrelevant to OER, as OER's optimized encodings do not depend on the Year property.

CENTURY

public static final int **CENTURY**

YEAR

public static final int **YEAR**

MONTH

public static final int **MONTH**

WEEK

public static final int **WEEK**

Week is present. When used for a duration (with DURATION), the following are excluded: CENTURY, YEAR, MONTH, DAY, HOURS, MINUTES, SECONDS When used for a date/datetime (without DURATION), MONTH is excluded.

DAY

public static final int **DAY**

(continued from last page)

HOURS

```
public static final int HOURS
```

MINUTES

```
public static final int MINUTES
```

SECONDS

```
public static final int SECONDS
```

UTC

```
public static final int UTC
```

Indicates a time is UTC. Its string format includes the 'Z' timezone designator.

DIFF

```
public static final int DIFF
```

Indicates a time includes a timezone difference from UTC. DIFF and UTC should never appear together.

FRACTION

```
public static final int FRACTION
```

This is a mask that is used to mark off the bits used for the number of decimal places for a fraction. value & FRACTION = # of decimal places

DURATION

```
public static final int DURATION
```

Indicates a duration. No other flags should be used with DURATION.

DT_ANY_CENTURY

```
public static final int DT_ANY_CENTURY
```

DT_ANY_YEAR

```
public static final int DT_ANY_YEAR
```

DT_ANY_YEAR_MONTH

```
public static final int DT_ANY_YEAR_MONTH
```

DT_ANY_YEAR_MONTH_DAY

```
public static final int DT_ANY_YEAR_MONTH_DAY
```

(continued from last page)

DT_ANY_YEAR_DAY

public static final int **DT_ANY_YEAR_DAY**

DT_ANY_YEAR_WEEK

public static final int **DT_ANY_YEAR_WEEK**

DT_ANY_YEAR_WEEK_DAY

public static final int **DT_ANY_YEAR_WEEK_DAY**

DT_CENTURY

public static final int **DT_CENTURY**

DT_YEAR

public static final int **DT_YEAR**

DT_YEAR_MONTH

public static final int **DT_YEAR_MONTH**

DT_YEAR_MONTH_DAY

public static final int **DT_YEAR_MONTH_DAY**

DT_YEAR_DAY

public static final int **DT_YEAR_DAY**

DT_YEAR_WEEK

public static final int **DT_YEAR_WEEK**

DT_YEAR_WEEK_DAY

public static final int **DT_YEAR_WEEK_DAY**

(continued from last page)

DT_HOURS

public static final int **DT_HOURS**

DT_HOURS_UTC

public static final int **DT_HOURS_UTC**

DT_HOURS_AND_DIFF

public static final int **DT_HOURS_AND_DIFF**

DT_MINUTES

public static final int **DT_MINUTES**

DT_MINUTES_UTC

public static final int **DT_MINUTES_UTC**

DT_MINUTES_AND_DIFF

public static final int **DT_MINUTES_AND_DIFF**

DT_SECONDS

public static final int **DT_SECONDS**

DT_SECONDS_UTC

public static final int **DT_SECONDS_UTC**

DT_SECONDS_AND_DIFF

public static final int **DT_SECONDS_AND_DIFF**

DT_HOURS_AND_FRACTION

public static final int **DT_HOURS_AND_FRACTION**

DT_HOURS_UTC_AND_FRACTION

public static final int **DT_HOURS_UTC_AND_FRACTION**

DT_HOURS_AND_DIFF_AND_FRACTION

public static final int **DT_HOURS_AND_DIFF_AND_FRACTION**

DT_MINUTES_AND_FRACTION

public static final int **DT_MINUTES_AND_FRACTION**

DT_MINUTES_UTC_AND_FRACTION

public static final int **DT_MINUTES_UTC_AND_FRACTION**

DT_MINUTES_AND_DIFF_AND_FRACTION

public static final int **DT_MINUTES_AND_DIFF_AND_FRACTION**

DT_SECONDS_AND_FRACTION

public static final int **DT_SECONDS_AND_FRACTION**

DT_SECONDS_UTC_AND_FRACTION

public static final int **DT_SECONDS_UTC_AND_FRACTION**

DT_SECONDS_AND_DIFF_AND_FRACTION

public static final int **DT_SECONDS_AND_DIFF_AND_FRACTION**

DATE

public static final int **DATE**

TIME_OF_DAY

public static final int **TIME_OF_DAY**

DATE_TIME

public static final int **DATE_TIME**

Constructors

(continued from last page)

Asn1TimeUtil

```
public Asn1TimeUtil()
```

Methods

convertToDER

```
public static java.lang.String convertToDER(java.lang.String timeValue)
```

Return the DER representation of a TimeValue. Given some TimeValue, this function returns a string representing the characters that should be encoded for that TimeValue when using DER. This can be invoked for values of type TIME. Its result is not correct for values of type DATE, TIME-OF-DAY, DATE-TIME, or DURATION, all of which require removing certain characters that are not removed for values of type TIME.

Parameters:

`timeValue`

Returns:

---

decodeDate

```
public static java.lang.String decodeDate(Asn1OerDecodeBuffer source,
         int flags)
  throws java.io.IOException
```

Decode a date value according to OER from the given source. This should be used for types having Basic="Date" and Date equal to "Y", "YM", or "YMD".

Parameters:

`source`
`flags`

Returns:

ASN.1 TimeValue representation of decoded value

---

decodeDateTime

```
public static java.lang.String decodeDateTime(Asn1OerDecodeBuffer source,
         int flags)
  throws java.io.IOException
```

Decode a date-time value from the given source. This should be used for types having Basic="Date-Time", Date equal to "Y", "YM", or "YMD", Time equal to "H", "HM", "HMS", or "HMSFn" and Local-or-UTC equal to "L", "Z", or "LD".

Parameters:

`source`
`flags` - Indicates what fields are expected. The flags should indicate a complete date (YMD, YWD, or YD) and at least an hour component.

Returns:

The value, represented as an ASN.1 TimeValue

---

decodeDuration

```
public static java.lang.String decodeDuration(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

(continued from last page)

Decode a duration according to OER from the given buffer. This should be used for types having Basic="Interval" and Interval-type="D"

Parameters:

`buffer`

Returns:

---

decodeTime

```
public static java.lang.String decodeTime(Asn1OerDecodeBuffer buffer,
         int flags)
  throws java.io.IOException
```

Decode a time value according to OER from the given source. This should be used for types meeting the following: 1) Basic="Time" 2) Time is one of "H", "HM", "HMS", "HMSFn" (for some fixed n) 3) Local-or-UTF is "L", "Z", or "LD"

Parameters:

`buffer`

`flags` - Indicates the expected fields.

Returns:

---

decodeDuration

```
public static java.lang.String decodeDuration(Asn1PerDecodeBuffer source,
         boolean recurring)
  throws Asn1Exception
```

Decode a (possibly recurring) duration from the given source. This should be used for types having Basic="Interval" or "Rec-Interval" and Interval-type="D"

Parameters:

`source`

`recurring` - true if the duration is a recurring duration

Returns:

---

decodeIntervalSE

```
public static java.lang.String decodeIntervalSE(Asn1PerDecodeBuffer source,
         boolean recurring,
         int flags)
  throws Asn1Exception
```

---

decodeIntervalSD

```
public static java.lang.String decodeIntervalSD(Asn1PerDecodeBuffer source,
         boolean recurring,
         int flags)
  throws Asn1Exception
```

(continued from last page)

decodeIntervalDE

```
public static java.lang.String decodeIntervalDE(Asn1PerDecodeBuffer source,
         boolean recurring,
         int flags)
  throws Asn1Exception
```

decodeDate

```
public static java.lang.String decodeDate(Asn1PerDecodeBuffer source,
         int flags)
  throws Asn1Exception
```

Decode a date value from the given source. This should be used for types having Basic="Date".

Parameters:

source

flags

Returns:

ASN.1 TimeValue representation of decoded value

decodeDateTime

```
public static java.lang.String decodeDateTime(Asn1PerDecodeBuffer source,
         int flags)
  throws Asn1Exception
```

Decode a date-time value from the given source. This should be used for types having Basic="Date-Time".

Parameters:

source

flags - Indicates what fields are expected. The flags should indicate a complete date (YMD, YWD, or YD) and at least an hour component.

Returns:

The value represented as an ASN.1 TimeValue

decodeTime

```
public static java.lang.String decodeTime(Asn1PerDecodeBuffer source,
         int flags)
  throws Asn1Exception
```

Decode a time value from the given source. This should be used for types having Basic="Time".

Parameters:

source

flags - Indicates the expected fields.

Returns:

encodeDate

```
public static void encodeDate(Asn1OerEncodeBuffer encoder,
         java.lang.String value,
         int flags)
  throws java.io.IOException
```

Encode a date value according to OER, using the given encoder. This should be used for types having Basic="Date" and Date equal to "Y", "YM" or "YMD".

(continued from last page)

Parameters:

encoder
value
flags - Must have YEAR set.

Throws:

Asn1Exception

---

encodeDateTime

```
public static void encodeDateTime(Asn1OerEncodeBuffer buffer,
          java.lang.String value,
          int flags)
  throws java.io.IOException
```

Encode a datetime value according to OER. This should be used for types having Basic="Date-Time", Date equal to "Y", "YM", or "YMD", Time equal to "H", "HM", "HMS", or "HMSFn" and Local-or-UTC equal to "L", "Z", or "LD".

Parameters:

buffer
value
flags

---

encodeDuration

```
public static void encodeDuration(Asn1OerEncodeBuffer buffer,
          java.lang.String value)
  throws java.io.IOException
```

Encode a duration according to OER. This should be used for types having Basic="Interval" and Interval-type="D".

Parameters:

buffer
value - The duration

---

encodeTime

```
public static void encodeTime(Asn1OerEncodeBuffer buffer,
          java.lang.String value,
          int flags)
  throws java.io.IOException
```

Encode a time value according to OER. This may be used only for TIME types that have been constrained so as to have an optimized encoding. This means being constrained as follows: 1) Basic=Time 2) Time is one of H, HM, HMS, HMSFn for some fixed n. 3) Local-or-UTF is one of L, Z, or LD

Parameters:

buffer
value
flags - Indicates the property constraints on the TIME type

Throws:

Asn1Exception

---

encodeDate

```
public static void encodeDate(Asn1PerEncoder encoder,
          java.lang.String value,
          int flags)
  throws Asn1Exception
```

Encode a date value according to PER, using the given encoder. This should be used for types having Basic="Date".

(continued from last page)

Parameters:

encoder
value
flags

Throws:

Asn1Exception

encodeDateTime

```
public static void encodeDateTime(Asn1PerEncoder encoder,
          java.lang.String value,
          int flags)
  throws Asn1Exception
```

Encode a datetime value. This should be used for types having Basic="Date-Time".

Parameters:

encoder
value
flags

Throws:

Asn1Exception

encodeDuration

```
public static void encodeDuration(Asn1PerEncoder encoder,
          java.lang.String value,
          boolean recurring)
  throws Asn1Exception
```

Encode a duration according to PER. This should be used for types having Basic="Interval" or "Rec-Interval", and Interval -type="D".

Parameters:

encoder
value - The duration
recurring - true if value represents a recurring duration.

encodeIntervalSE

```
public static void encodeIntervalSE(Asn1PerEncoder encoder,
          java.lang.String value,
          boolean recurring,
          int flags)
  throws Asn1Exception
```

encodeIntervalSD

```
public static void encodeIntervalSD(Asn1PerEncoder encoder,
          java.lang.String value,
          boolean recurring,
          int flags)
  throws Asn1Exception
```

(continued from last page)

encodeIntervalDE

```
public static void encodeIntervalDE(Asn1PerEncoder encoder,
         java.lang.String value,
         boolean recurring,
         int flags)
  throws Asn1Exception
```

---

encodeTime

```
public static void encodeTime(Asn1PerEncoder encoder,
         java.lang.String value,
         int flags)
  throws Asn1Exception
```

Encode a time value. This should be used for types having Basic="Time".

Parameters:

encoder
value
flags

Throws:

Asn1Exception

com.objsys.asn1j.runtime
Class Asn1TraceHandler

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1TraceHandler
```

All Implemented Interfaces:
Asn1NamedEventHandler

---

public class **Asn1TraceHandler**
extends java.lang.Object
implements Asn1NamedEventHandler

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).

Constructor Summary

public	`Asn1TraceHandler()` This constructor sets the output stream to standard output.
public	`Asn1TraceHandler(java.io.PrintStream ps)` This constructor sets the output stream to the given PrintStream.

Method Summary

void	`characters(java.lang.String svalue,short typeCode)` The characters callback method is invoked when content (primitive data) is encountered.
void	`endElement(java.lang.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.
void	`startElement(java.lang.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.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1TraceHandler

`public` **`Asn1TraceHandler`**`()`

This constructor sets the output stream to standard output.

Asn1TraceHandler

`public` **`Asn1TraceHandler`**`(java.io.PrintStream ps)`

This constructor sets the output stream to the given PrintStream.

Methods

characters

```
public void characters(java.lang.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).

startElement

```
public void startElement(java.lang.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.

endElement

```
public void endElement(java.lang.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.

com.objsys.asn1j.runtime
Class Asn1Type

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Type
```

All Implemented Interfaces:
java.lang.Cloneable, java.io.Serializable, Asn1TypeIF

Direct Known Subclasses:
Asn1BigInteger, Asn1X694OrderElement, Asn1UniversalString, Asn1SeqSetOf, Asn1SeqOrSet, Asn1Real, Asn1OpenExt, Asn1OctetString, Asn1ObjectIdentifier, Asn1Null, Asn1Integer, Asn1Enumerated, Asn1Choice, Asn1CharString, Asn1Boolean, Asn1BitString, Asn1ArrayType

---

public abstract class **Asn1Type**
extends java.lang.Object
implements Asn1TypeIF, java.io.Serializable, java.lang.Cloneable

This is the base class for all ASN.1 built-in types.

Field Summary

static final short	**BIT_STRING** BIT_STRING type code = 3 Value: **3**
static final short	**BMPString** BMPString type code = 30 Value: **30**
static final short	**BOOLEAN** BOOLEAN type code = 1 Value: **1**
static final short	**DATE** DATE type code = 31 Value: **31**
static final short	**DATE_TIME** DATE-TIME type code = 33 Value: **33**
static final short	**DURATION** DURATION type code = 34 Value: **34**
static final short	**ENUMERATED** ENUMERATED type code = 10 Value: **10**

static final short	EOC EOC type code = 0 Value: **0**
static final short	EXTERNAL EXTERNAL type code = 8 Value: **8**
static final short	GeneralString GeneralString type code = 27 Value: **27**
static final short	GeneralTime GeneralTime type code = 24 Value: **24**
static final short	GraphicString GraphicString type code = 25 Value: **25**
static final short	IA5String IA5String type code = 22 Value: **22**
static final short	INTEGER INTEGER type code = 2 Value: **2**
java.lang.String	mNonParameterizedTypeName This type's NonParameterizedTypeName.
static final short	NULL NULL type code = 5 Value: **5**
static final short	NumericString NumericString type code = 18 Value: **18**
static final short	OBJECT_IDENTIFIER OBJECT_IDENTIFIER type code = 6 Value: **6**
static final short	ObjectDescriptor ObjectDescriptor type code = 7 Value: **7**
static final short	OCTET_STRING OCTET_STRING type code = 4 Value: **4**

static final short	`OID_IRI` OID-IRI type code = 35 Value: **35**
static final short	`OpenType` OpenType type code = 99 Value: **99**
static final short	`PrintableString` PrintableString type code = 19 Value: **19**
static final short	`REAL` REAL type code = 9 Value: **9**
static final short	`RELATIVE_OID_IRI` RELATIVE-OID-IRI type code = 35 Value: **36**
static final short	`RelativeOID` RELATIVE_OID type code = 13 Value: **13**
static final short	`SEQUENCE` SEQUENCE type code = 16 Value: **16**
static final short	`SET` SET type code = 17 Value: **17**
static final short	`T61String` T61String type code = TeletexString Value: **20**
static final short	`TeletexString` TeletexString type code = 20 Value: **20**
static final short	`TIME` TIME type code = 14 Value: **14**
static final short	`TIME_OF_DAY` TIME-OF-DAY type code = 32 Value: **32**
static final short	`UniversalString` type code= 28 Value: **28**

static final short	UTCTime UTCTime type code = 23 Value: **23**
static final short	UTF8String UTF8String type code = 12 Value: **12**
static final short	VideotexString VideotexString type code = 21 Value: **21**
static final short	VisibleString VisibleString type code = 26 Value: **26**

Constructor Summary

public	Asn1Type()

Method Summary

static void	_setKey(byte[] rtkey)
java.lang.Object	clone() Creates and returns a copy of this object.
void	decode(Asn1BerDecodeBuffer buffer) This method is used to decode a message that is encoded in BER or DER format.
static Asn1Type	decode(Asn1BerDecodeBuffer buffer,Asn1OpenTypeField openTypeField,boolean explicit,int implicitLength) Decode an open type field value from the given buffer.
void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method is used to decode a message that is encoded in BER or DER format.
void	decode(Asn1JsonDecodeBuffer buffer) Decode this type from the given buffer.
static Asn1Type	decode(Asn1JsonDecodeBuffer buffer,Asn1OpenTypeField openTypeField) Decode an open type field value from the given buffer.
void	decode(Asn1MderDecodeBuffer buffer) This method decodes this object's data from an MDER encoding.
void	decode(Asn1NasDecodeBuffer buffer) This method is the base implementation of the NAS decode method.

static Asn1Type	`decode(Asn1NasDecodeBuffer buffer,Asn1OpenTypeField openTypeField)` Decode an open type field value from the given buffer.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 value into this object, following the Octet Encoding Rules (OER).
static Asn1Type	`decode(Asn1OerDecodeBuffer buffer,Asn1OpenTypeField openTypeField)` Decode an open type field value from the given buffer.
void	`decode(Asn1PerDecodeBuffer buffer)` This method is the base implementation of the standard Packed Encoding Rules (PER) decode method.
static Asn1Type	`decode(Asn1PerDecodeBuffer buffer,Asn1OpenTypeField openTypeField)` Decode an open type field value from the given buffer.
void	`decode(java.lang.Object reader,java.io.InputStream byteStream)` This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.
void	`decode(java.lang.Object reader,java.lang.String xmlURI)` This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes the XML content of a simple type.
int	`encode(Asn1BerEncodeBuffer buffer)` This method is used to encode a message in BER or DER format.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method is used to encode this data type in BER or DER format.
void	`encode(Asn1BerOutputStream out,boolean explicit)` This method writes to the stream an encoded ASN.1 type value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`encode(Asn1JsonOutputStream outstream)` Encode this type to the given output stream.
void	`encode(Asn1MderOutputStream buffer)` This method encodes this object's data to an MDER encoding.
void	`encode(Asn1MderOutputStream buffer,boolean useCachedLength)` This method encodes this object's data to an MDER encoding.
void	`encode(Asn1NasEncodeBuffer buffer)` This method is the base implementation of the NAS encode method.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes the ASN.1 value represented by this object, following the Octet Encoding Rules (OER).

void	`encode(Asn1PerEncodeBuffer buffer)` This method is the base implementation of the standard Packed Encoding Rules (PER) encode method.
void	`encode(Asn1PerOutputStream out)` This method is the base implementation of the standard Packed Encoding Rules (PER) encode method using output stream.
void	`encode(Asn1XerEncoder buffer)` This method is the base implementation of the standard XML Encoding Rules (XER) encode method.
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method is the base implementation of the standard XML Encoding Rules (XER) encode method.
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method is the base implementation of the standard XML Encoding as specified in the XML schema standard(asn2xsd).
void	`encodeAsOpenType(Asn1OerEncodeBuffer buffer)` This method encodes the ASN.1 value represented by this object, following the Octet Encoding Rules (OER), as the value for the actual type of an open type (i.e., it precedes its encoding with a length).
boolean	`equals(Asn1Type obj)` Return true if the two objects are equal.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 specification type name for this type.
int	`getLength()` This method will return the length of types that can be bound by a size constraint (BIT STRING, OCTET STRING, character string, and SEQUENCE OF/SET OF).
java.lang.String	`getNonParameterizedTypeName()` Return this type's NonParameterizedTypeName, if set.
static java.lang.String	`getTypeName(short typeCode)` The static getTypeName method will convert a type code into a type name as defined in the X.680 standard..
int	`hashCode()` Returns the hash code of the object.
void	`indent(java.io.PrintWriter out,int level)` This method will indent three spaces in the given print stream.
void	`indent(java.lang.StringBuilder sb,int level)` This method will indent three spaces in the given string builder.
boolean	`isOpenType()` Returns open type mode for XML encoding/decoding.

static int	matchTag(Asn1BerDecodeBuffer buffer,Asn1Tag tag) This method will compare the next parsed tag with the given tag value.
static int	matchTag(Asn1BerDecodeBuffer buffer,short tagClass,short tagForm,int tagIDCode) This method will compare the next parsed tag with the given tag value.
void	pdiag(java.lang.String s) This is a diagnostics print method.
void	print(java.io.PrintStream out,java.lang.String varName,int level) This method will format and output a primitive value to the given print stream.
void	print(java.io.PrintWriter out,java.lang.String varName,int level) This method will format and output a primitive value to the given print writer.
void	print(java.lang.StringBuilder sb,java.lang.String varName,int level) This method will format and output a primitive value to the given string builder.
void	setNonParameterizedTypeName(java.lang.String value) Set this type's NonParameterizedTypeName.
void	setOpenType() Sets open type mode for XML encoding/decoding.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

EOC

public static final short **EOC**

EOC type code = 0

BOOLEAN

public static final short **BOOLEAN**

BOOLEAN type code = 1

INTEGER

public static final short **INTEGER**

INTEGER type code = 2

BIT_STRING

public static final short **BIT_STRING**

BIT_STRING type code = 3

OCTET_STRING

```
public static final short OCTET_STRING
```

OCTET_STRING type code = 4

NULL

```
public static final short NULL
```

NULL type code = 5

OBJECT_IDENTIFIER

```
public static final short OBJECT_IDENTIFIER
```

OBJECT_IDENTIFIER type code = 6

ObjectDescriptor

```
public static final short ObjectDescriptor
```

ObjectDescriptor type code = 7

EXTERNAL

```
public static final short EXTERNAL
```

EXTERNAL type code = 8

REAL

```
public static final short REAL
```

REAL type code = 9

ENUMERATED

```
public static final short ENUMERATED
```

ENUMERATED type code = 10

UTF8String

```
public static final short UTF8String
```

UTF8String type code = 12

RelativeOID

```
public static final short RelativeOID
```

RELATIVE_OID type code = 13

TIME

```
public static final short TIME
```

TIME type code = 14

(continued from last page)

SEQUENCE

`public static final short SEQUENCE`

SEQUENCE type code = 16

SET

`public static final short SET`

SET type code = 17

NumericString

`public static final short NumericString`

NumericString type code = 18

PrintableString

`public static final short PrintableString`

PrintableString type code = 19

TeletexString

`public static final short TeletexString`

TeletexString type code = 20

T61String

`public static final short T61String`

T61String type code = TeletexString

VideotexString

`public static final short VideotexString`

VideotexString type code = 21

IA5String

`public static final short IA5String`

IA5String type code = 22

UTCTime

`public static final short UTCTime`

UTCTime type code = 23

GeneralTime

`public static final short GeneralTime`

GeneralTime type code = 24

GraphicString

`public static final short GraphicString`

GraphicString type code = 25

VisibleString

```
public static final short VisibleString
```

VisibleString type code = 26

GeneralString

```
public static final short GeneralString
```

GeneralString type code = 27

UniversalString

```
public static final short UniversalString
```

type code= 28

BMPString

```
public static final short BMPString
```

BMPString type code = 30

DATE

```
public static final short DATE
```

DATE type code = 31

TIME_OF_DAY

```
public static final short TIME_OF_DAY
```

TIME-OF-DAY type code = 32

DATE_TIME

```
public static final short DATE_TIME
```

DATE-TIME type code = 33

DURATION

```
public static final short DURATION
```

DURATION type code = 34

OID_IRI

```
public static final short OID_IRI
```

OID-IRI type code = 35

RELATIVE_OID_IRI

```
public static final short RELATIVE_OID_IRI
```

RELATIVE-OID-IRI type code = 35

(continued from last page)

OpenType

```
public static final short OpenType
```

OpenType type code = 99

---

mNonParameterizedTypeName

```
protected transient java.lang.String mNonParameterizedTypeName
```

This type's NonParameterizedTypeName.

Constructors

Asn1Type

```
public Asn1Type()
```

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method is used to decode a message that is encoded in BER or DER format.

Parameters:

`buffer` - Decode message buffer object
`explicit` - Flag indicating explicit tag should be parsed from the encoded type.
`implicitLength` - Length of the contents field (only required if explicit is false).

---

decode

```
public void decode(Asn1BerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

decode

```
public static Asn1Type decode(Asn1BerDecodeBuffer buffer,
         Asn1OpenTypeField openTypeField,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

Decode an open type field value from the given buffer.

Parameters:

`buffer`
`openTypeField` - Describes the open type being decoded.

(continued from last page)

`explicit` - if `true`, the value to be decoded is preceded by a tag and length, which must first be decoded. Otherwise, implicitLength provides the length of the value to be decoded.
`implicitLength` - The length of the value to be decoded if explicit was given as `false`.

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

This method is used to encode this data type in BER or DER format.

Parameters:

`buffer` - Encode message buffer object
`explicit` - Flag indicating explicit tag should be added to the encoded type.

Returns:

Decoded integer value

encode

```
public int encode(Asn1BerEncodeBuffer buffer)
  throws Asn1Exception
```

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:

Decoded integer value

decode

```
public static Asn1Type decode(Asn1OerDecodeBuffer buffer,
         Asn1OpenTypeField openTypeField)
  throws Asn1Exception,
         java.io.IOException
```

Decode an open type field value from the given buffer.

Parameters:

`buffer`
`openTypeField` - Describes the open type being decoded.

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 value into this object, following the Octet Encoding Rules (OER). This method MUST be overridden by all subclasses, except for Asn1Enumerated and subclasses thereof, in order to implement the correct decode behavior for the corresponding ASN.1 type. This method MUST NOT be invoked on an Asn1Enumerated object. Such objects are immutable and cannot be decoded into. This method is used polymorphically when table constraint code is generated and Asn1Type is used as the declared type for an open type.

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

(continued from last page)

This method encodes the ASN.1 value represented by this object, following the Octet Encoding Rules (OER). This method MUST be overridden by all subclasses in order to implement the correct encode behavior for the corresponding type. This method is used polymorphically when table constraint code is generated and Asn1Type is used as the declared type for an open type.

Parameters:

`buffer` - Encode message buffer object

---

encodeAsOpenType

```
public final void encodeAsOpenType(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes the ASN.1 value represented by this object, following the Octet Encoding Rules (OER), as the value for the actual type of an open type (i.e., it precedes its encoding with a length).

Parameters:

`buffer` - Encode message buffer object

---

decode

```
public static Asn1Type decode(Asn1NasDecodeBuffer buffer,
         Asn1OpenTypeField openTypeField)
  throws java.io.IOException
```

Decode an open type field value from the given buffer.

Parameters:

`buffer`

`openTypeField` - Describes the actual type for the open type.

---

decode

```
public void decode(Asn1NasDecodeBuffer buffer)
  throws java.io.IOException
```

This method is the base implementation of the NAS decode method. It throws an exception because it should never be invoked by compiler generated code.

Parameters:

`buffer` - NAS decode buffer object

---

decode

```
public static Asn1Type decode(Asn1PerDecodeBuffer buffer,
         Asn1OpenTypeField openTypeField)
  throws Asn1Exception,
         java.io.IOException
```

Decode an open type field value from the given buffer.

Parameters:

`buffer`

`openTypeField` - Describes the actual type for the open type.

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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.

(continued from last page)

Parameters:

`buffer` - PER Encode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

encode

```
public void encode(Asn1NasEncodeBuffer buffer)
  throws java.io.IOException
```

This method is the base implementation of the NAS encode method. This throws an exception because it should never be invoked by compiler generated code.

Parameters:

`buffer` - NAS Encode message buffer object

---

encode

```
public void encode(Asn1XmlEncoder buffer,
          java.lang.String elemName,
          java.lang.String nsPrefix)
  throws Asn1Exception,
         java.io.IOException
```

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
`elemName` - XML element name of item
`nsPrefix` - Element namespace prefix value

Throws:

`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1XerEncoder buffer)
  throws Asn1Exception,
         java.io.IOException
```

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.

Parameters:

`buffer` - XER Encode message buffer object

Throws:

`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws Asn1Exception,
         java.io.IOException
```

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.

Parameters:

`buffer` - XER Encode message buffer object
`elemName` - XML element name of item

Throws:

`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

decode

```
public void decode(java.lang.Object reader,
         java.lang.String xmlURI)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.

Parameters:

`reader` - XML reader object
`xmlURI` - URI of a source

Throws:

`IOException` - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.
`Asn1Exception` - Thrown, if operation is failed.

decode

```
public void decode(java.lang.Object reader,
         java.io.InputStream byteStream)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.

Parameters:

`reader` - XML reader object
`byteStream` - Input byte stream object

Throws:

`IOException` - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.
`Asn1Exception` - Thrown, if operation is failed.

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes the XML content of a simple type.

Parameters:

(continued from last page)

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

setOpenType

`public void setOpenType()`

Sets open type mode for XML encoding/decoding.

isOpenType

`public boolean isOpenType()`

Returns open type mode for XML encoding/decoding.

Returns:

true if open type mode is on.

decode

```
public void decode(Asn1MderDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes this object's data from an MDER encoding. The implementation for this class throws an exception. When generating MDER code, subclasses will override this implementation.

Parameters:

`buffer` - MDER Decode message buffer object

Throws:

`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1MderOutputStream buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes this object's data to an MDER encoding. The implementation for this class invokes encode(buffer, false). When generating MDER code, subclasses will override this implementation or else will override the encode(Asn1MderOutputStream, boolean) overload. Invoke this method if you are not certain which overload has been overridden by the subclass.

Parameters:

`buffer` - MDER Encode message buffer object

Throws:

`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1MderOutputStream buffer,
          boolean useCachedLength)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes this object's data to an MDER encoding. The implementation for this class throws an exception. When generating MDER code, subclasses will override this implementation or else will override the encode(Asn1MderOutputStream) overload. Do not invoke this function unless you are certain the subclass has overridden it.

(continued from last page)

Parameters:

`buffer` - MDER Encode message buffer object

`useCachedLength` - Indicates whether cached length of encoding should be used. In general, only generated code should pass true.

Throws:

`IOException` - Any exception thrown by the underlying stream.

`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode this type from the given buffer. The implementation for this class throws an exception. Do not invoke this function unless you are certain the subclass has overridden it.

Parameters:

`buffer` - JSON decode buffer

---

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this type to the given output stream. The implementation for this class throws an exception. Do not invoke this function unless you are certain the subclass has overridden it.

Parameters:

`outstream` - JSON output stream

---

decode

```
public static Asn1Type decode(Asn1JsonDecodeBuffer buffer,
         Asn1OpenTypeField openTypeField)
  throws Asn1Exception,
         java.io.IOException
```

Decode an open type field value from the given buffer.

Parameters:

`buffer`

`openTypeField` - Describes the open type being decoded.

---

getLength

```
public int getLength()
  throws Asn1InvalidLengthException
```

This method will return 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.

Returns:

Length of item in units (for example, bits for BIT STRING, octets for OCTET STRING, etc.)

_setKey

```
public static void _setKey(byte[] rtkey)
```

(continued from last page)

getNonParameterizedTypeName

```
public java.lang.String getNonParameterizedTypeName()
```

Return this type's NonParameterizedTypeName, if set.

setNonParameterizedTypeName

```
public void setNonParameterizedTypeName(java.lang.String value)
```

Set this type's NonParameterizedTypeName. The value is specified, based on the ASN.1 type, by X.680.

matchTag

```
protected static int matchTag(Asn1BerDecodeBuffer buffer,
         short tagClass,
         short tagForm,
         int tagIDCode)
  throws Asn1Exception,
         java.io.IOException
```

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

Throws:

`Asn1TagMatchFailedException` - Tag is not equal to expected value

matchTag

```
protected static int matchTag(Asn1BerDecodeBuffer buffer,
         Asn1Tag tag)
  throws Asn1Exception,
         java.io.IOException
```

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

Throws:

`Asn1TagMatchFailedException` - Tag is not equal to expected value

indent

```
public void indent(java.io.PrintWriter out,
         int level)
```

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:

(continued from last page)

`out` - Print stream
`level` - Indentation level (# of spaces is 3 x this number)

indent

```
public void indent(java.lang.StringBuilder sb,
          int level)
```

This method will indent three spaces in the given string builder. It is used by print to string methods to provide a formatted output of an encoded element value.

Parameters:

`sb` - The output string builder.
`level` - The indentation level; the number of spaces is three times this number.

pdiag

```
public void pdiag(java.lang.String s)
```

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.

print

```
public void print(java.io.PrintStream out,
          java.lang.String varName,
          int level)
```

This method will format and output a primitive value to the given print stream.

Parameters:

`out` - Print output stream
`varName` - Name of variable
`level` - Indentation level

print

```
public void print(java.io.PrintWriter out,
          java.lang.String varName,
          int level)
```

This method will format and output a primitive value to the given print writer.

Parameters:

`out` - Print writer for output
`varName` - Name of variable
`level` - Indentation level

print

```
public void print(java.lang.StringBuilder sb,
          java.lang.String varName,
          int level)
```

This method will format and output a primitive value to the given string builder. Synchronization is left to the caller.

Parameters:

`sb` - The String Builder used for output.
`varName` - The variable name.
`level` - The indentation level.

getTypeName

```
public static java.lang.String getTypeName(short typeCode)
```

The static getTypeName method will convert a type code into a type name as defined in the X.680 standard..

Parameters:

`typeCode` - Type code to be converted

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

clone

```
public java.lang.Object clone()
  throws java.lang.CloneNotSupportedException
```

Creates and returns a copy of this object.

Returns:

a clone of this instance.

Throws:

`CloneNotSupportedException` - if the object's class does not support the `Cloneable`interface. Subclasses that override the `clone`method can also throw this exception to indicate that an instance cannot be cloned.

See Also:

`java.lang.Cloneable`

(continued from last page)

equals

`public boolean **equals**(Asn1Type obj)`

Return true if the two objects are equal. Note: The default implementation here is to invoke equals(Object). This is acceptable for the runtime classes, which have overridden equals(Object).

hashCode

`public int **hashCode**()`

Returns the hash code of the object. Note: The default behavior of the hashcode is to return 1. This method is overridden in subclasses of Asn1Type to produce real hashcodes.

getAsn1TypeName

`public java.lang.String **getAsn1TypeName**()`

Returns the ASN.1 specification type name for this type.

com.objsys.asn1j.runtime
Interface Asn1TypeIF

All Known Implementing Classes:
Asn1Type

---

public interface **Asn1TypeIF**

This is the base interface for all ASN.1 built-in types.

Method Summary

abstract void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method declaration is the signature of the standard Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) decode method.
abstract void	`decode(Asn1MderDecodeBuffer buffer)` This method decodes this object's data from an MDER encoding.
abstract void	`decode(Asn1PerDecodeBuffer buffer)` This method declaration is the signature of the standard Packed Encoding Rules (PER) decode method.
abstract void	`decode(java.lang.Object reader,java.io.InputStream byteStream)` This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.
abstract void	`decode(java.lang.Object reader,java.lang.String xmlURI)` This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.
abstract void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes the XML content of a simple type.
abstract int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method declaration is the signature of the standard Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) encode method.
abstract void	`encode(Asn1BerOutputStream out,boolean explicit)` This method declaration is the signature of the streaming oriented BER encode method.
abstract void	`encode(Asn1MderOutputStream buffer)` This method encodes this object's data to an MDER encoding.
abstract void	`encode(Asn1PerEncodeBuffer buffer)` This method declaration is the signature of the standard Packed Encoding Rules (PER) encode method.
abstract void	`encode(Asn1PerOutputStream out)` This method declaration is the signature of the streaming oriented PER encode method.

abstract void	encode(Asn1XerEncoder buffer) This method declaration is the signature of the standard XML Encoding Rules (XER) encode method.
abstract void	encode(Asn1XerEncoder buffer,java.lang.String elemName) This method declaration is the signature of the standard XML Encoding Rules (XER) encode method.
abstract boolean	isOpenType() Returns open type mode for XML encoding/decoding.
abstract void	print(java.io.PrintStream out,java.lang.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.
abstract void	print(java.io.PrintWriter out,java.lang.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.
abstract void	print(java.lang.StringBuilder out,java.lang.String varName,int level) This method declaration is the signature of the standard print to string method used to print the contents of the object representing the ASN.1 type.
abstract void	setOpenType() Sets open type mode for XML encoding/decoding.

Methods

decode

```
public abstract void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating explicit tag should be parsed from the encoded type.
`implicitLength` - Length of the contents field (only required if explicit is false).

encode

```
public abstract int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tag should be added to the encoded type.

(continued from last page)

Returns:
Decoded integer value

---

decode

```
public abstract void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard Packed Encoding Rules (PER) decode method.

Parameters:
`buffer` - PER Encode message buffer object

---

encode

```
public abstract void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard Packed Encoding Rules (PER) encode method.

Parameters:
`buffer` - PER Encode message buffer object

---

encode

```
public abstract void encode(Asn1XerEncoder buffer)
  throws Asn1Exception,
         java.io.IOException
```

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.

Parameters:
`buffer` - XER Encode message buffer object

Throws:
`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public abstract void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard XML Encoding Rules (XER) encode method.

Parameters:
`buffer` - XER Encode message buffer object
`elemName` - XML element name of item

Throws:
`IOException` - Any exception thrown by the underlying stream.
`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

decode

```
public abstract void decode(java.lang.Object reader,
         java.lang.String xmlURI)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.

Parameters:

`reader` - XML reader object
`xmlURI` - URI of a source

Throws:

`IOException` - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.
`Asn1Exception` - Thrown, if operation is failed.

---

decode

```
public abstract void decode(java.lang.Object reader,
         java.io.InputStream byteStream)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the standard XML Encoding Rules (XER) decode method.

Parameters:

`reader` - XML reader object
`byteStream` - Input byte stream object

Throws:

`IOException` - An IO exception from the parser, possibly from a byte stream or character stream supplied by the application.
`Asn1Exception` - Thrown, if operation is failed.

---

decodeXML

```
public abstract void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes the XML content of a simple type.

Parameters:

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

---

setOpenType

```
public abstract void setOpenType()
```

Sets open type mode for XML encoding/decoding.

---

isOpenType

```
public abstract boolean isOpenType()
```

Returns open type mode for XML encoding/decoding.

Returns:

true if open type mode is on.

decode

```
public abstract void decode(Asn1MderDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes this object's data from an MDER encoding. When MDER code has not been generated, classes implementing this interface may simply throw an exception.

encode

```
public abstract void encode(Asn1MderOutputStream buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes this object's data to an MDER encoding. When MDER code has not been generated, classes implementing this interface may simply throw an exception.

encode

```
public abstract void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the streaming oriented BER encode method.

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public abstract void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

This method declaration is the signature of the streaming oriented PER encode method.

Parameters:

`out` - BER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.

`Asn1Exception` - Thrown, if operation is failed.

print

```
public abstract void print(java.io.PrintWriter out,
         java.lang.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. This method is provided for backwards-compatibility with earlier versions of the runtime; users should instead use a PrintWriter to obtain better output performance.

Parameters:

`out` - Output print writer

`varName` - Name of the variable being printed

(continued from last page)

`level` - Indentation level

print

```
public abstract void print(java.io.PrintStream out,
          java.lang.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. This method is provided for backwards-compatibility with earlier versions of the runtime; users should instead use a PrintWriter to obtain better output performance.

Parameters:

`out` - Output print stream
`varName` - Name of the variable being printed
`level` - Indentation level

print

```
public abstract void print(java.lang.StringBuilder out,
          java.lang.String varName,
          int level)
```

This method declaration is the signature of the standard print to string method used to print the contents of the object representing the ASN.1 type. N.b., StringBuilder is not synchronized, and calls to this method from a multiple threads must be synchronized by the caller.

Parameters:

`out` - Output string builder
`varName` - Name of the variable being printed
`level` - Indentation level

com.objsys.asn1j.runtime
Class Asn1UnexpectedElementException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-java.lang.RuntimeException
            |
            +-com.objsys.asn1j.runtime.Asn1Exception
               |
               +-com.objsys.asn1j.runtime.Asn1UnexpectedElementException
```

public class **Asn1UnexpectedElementException**
extends Asn1Exception

This class defines the ASN.1 unexpected element exception that is thrown when an unexpected element is detected in a SEQUENCEor SETtype.

This exception may be thrown when an element is decoded in a SEQUENCEor SETthat was not specified in the input grammar. This is distinguished from the Asn1SeqOrderException, which is thrown when elements are decoded out of order.

Constructor Summary

public	`Asn1UnexpectedElementException()` This constructor creates an exception object with a default textual message.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1UnexpectedElementException

`public` **`Asn1UnexpectedElementException`**`()`

This constructor creates an exception object with a default textual message.

com.objsys.asn1j.runtime
Class Asn1UniversalString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1UniversalString
```

public class **Asn1UniversalString**
extends Asn1Type

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

Field Summary

static final int	BITSPERCHAR The **BITSPERCHAR**constant specifies the number of bits per character for PER (aligned or unaligned). Value: **32**
java.lang.StringBuffer	mStringBuffer
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 28).
int[]	value The **value**public member variable is used to hold the string value to be encoded or the results of a decode operation.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1UniversalString() This constructor creates an empty string that can be used in a decode method call to receive a string value.
public	Asn1UniversalString(int[] data) This constructor initializes the character string from the given universal string data.
public	Asn1UniversalString(java.lang.String data) This constructor converts a standard Java string value into a universal string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 universal string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 restricted character string from JSON.
void	`decode(Asn1OerDecodeBuffer buffer)` Decode the value in accordance with OER.
void	`decode(Asn1OerDecodeBuffer buffer,int length)` Decode the value in accordance with OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,Asn1CharSet charSet)` This method decodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet)` This method decodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,int nchars,int abpc,int ubpc,Asn1CharSet charSet,int startIdx)` This method decodes the contents of a **UniversalString**.
void	`decodeXER(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 Universal String value using the XML encoding rules (XER).
void	`decodeXML(java.lang.String buffer,java.lang.String attrs)` This method decodes an ASN.1 Universal String value using the XML schema encoding rules.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 universal string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.

void	`encode(Asn1JsonOutputStream out)` Encode the value of this object as a JSON string.
void	`encode(Asn1OerEncodeBuffer buffer)` Encode the value in accordance with OER.
void	`encode(Asn1OerEncodeBuffer buffer,boolean withLength)` Encode the string, with or without a length determinant.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,Asn1CharSet charSet)` This method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,int abpc,int ubpc,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,int nchars,int offset,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes the contents of a **UniversalString**type.
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,Asn1CharSet charSet)` This method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out,int abpc,int ubpc,Asn1CharSet charSet,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 **UniversalString**value in accordance with the packed encoding rules (PER).

void	`encode(Asn1PerOutputStream out,int nchars,int offset,int abpc,int ubpc,Asn1CharSet charSet)` This method encodes the contents of a **UniversalString**type.
void	`encode(Asn1XerEncoder buffer,java.lang.String elemName)` This method encodes an ASN.1 Universal String value using the XML encoding rules (XER).
void	`encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an ASN.1 Universal String value with element and attribute name tag using the XML Encoding as specified in the XML schema standard(asn2xsd).
void	`encodeData(Asn1XmlXerEncoder buffer)` This method encodes an ASN.1 Universal String value using the XML Encoding as specified in the XML schema standard(asn2xsd).
boolean	`equals(java.lang.Object cs_)` This method compares this character string value to the given value for equality.
java.lang.String	`getAsn1TypeName()` Returns the ASN.1 type name.
int	`getLength()` This method will return the length of the character string in characters.
int	`hashCode()` This method returns the hashcode for this object.
java.lang.String	`toString()` This method will return a string representation of the value.
boolean	`validate(Asn1CharSet charSet)` This method will attempt to validate a string against its internal character set.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

(continued from last page)

BITSPERCHAR

`public static final int BITSPERCHAR`

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

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag TAG`

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 28).

---

value

`public transient int value`

The **value**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.

---

mStringBuffer

`protected transient java.lang.StringBuffer mStringBuffer`

Constructors

Asn1UniversalString

`public Asn1UniversalString()`

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

Asn1UniversalString

`public Asn1UniversalString(int[] data)`

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

Parameters:

`data` - Character string

Asn1UniversalString

`public Asn1UniversalString(java.lang.String data)`

This constructor converts a standard Java string value into a universal string. If a null string is passed, the constructor produces a zero-length Universal String.

Parameters:

`data` - Character string

Methods

getAsn1TypeName

`public java.lang.String getAsn1TypeName()`

Returns the ASN.1 type name.

Returns:

(continued from last page)

The ASN.1 type name UniversalString.

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:

`buffer` - Decode message buffer object

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:

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

(continued from last page)

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         Asn1CharSet charSet,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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

---

decode

```
protected void decode(Asn1PerDecodeBuffer buffer,
         int abpc,
         int ubpc,
         Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**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)

---

decode

```
protected void decode(Asn1PerDecodeBuffer buffer,
         int nchars,
         int abpc,
         int ubpc,
         Asn1CharSet charSet,
         int startIdx)
  throws Asn1Exception,
         java.io.IOException
```

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

(continued from last page)

decode

```
protected void decode(Asn1PerDecodeBuffer buffer,
          int abpc,
          int ubpc,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**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

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**member variable.

Parameters:

`buffer` - Encode message buffer object

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          Asn1CharSet charSet)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**member variable.

Parameters:

`buffer` - Encode message buffer object
`charSet` - Object representing permitted alphabet constraint character set (optional)

---

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**member variable.

Parameters:

(continued from last page)

`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

encode

```
protected void encode(Asn1PerEncodeBuffer buffer,
          int nchars,
          int offset,
          int abpc,
          int ubpc,
          Asn1CharSet charSet)
  throws Asn1Exception
```

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)

encode

```
protected void encode(Asn1PerEncodeBuffer buffer,
          int abpc,
          int ubpc,
          Asn1CharSet charSet)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**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)

encode

```
protected void encode(Asn1PerEncodeBuffer buffer,
          int abpc,
          int ubpc,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**member variable.

Parameters:

`buffer` - Encode message buffer object
`abpc` - Number of bits per character (aligned)
`ubpc` - Number of bits per character (unaligned)

(continued from last page)

`charSet` - Object representing the permitted alphabet constraint character set (optional)
`lower` - Effective size constraint lower bound
`upper` - Effective size constraint upper bound

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode the value in accordance with OER. This class's implementation decodes the string with a length determinant. If a subclass should be decoded without a length determinant, it should override this method to invoke decode(buffer, length). Subclasses may override this to add constraint checks after invoking this method to decode the string.

Parameters:
`buffer`

decode

```
public final void decode(Asn1OerDecodeBuffer buffer,
         int length)
  throws java.io.IOException
```

Decode the value in accordance with OER. This class's implementation decodes a string of the given length. This method is final as I don't see any reason for overriding it.

Parameters:
`buffer`
`length` - Length of string to decode, in characters.

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode the value in accordance with OER. This class's implementation invokes encode(buffer, true) to encode the string with a length determinant. If a subclass should be encoded without a length determinant, it should override this to invoke encode(buffer, false).

Parameters:
`buffer`

encode

```
public void encode(Asn1OerEncodeBuffer buffer,
         boolean withLength)
  throws java.io.IOException
```

Encode the string, with or without a length determinant. Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters:
`buffer`
`withLength` - true if a length determinant should be encoded.

encode

```
public void encode(Asn1XerEncoder buffer,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

(continued from last page)

`buffer` - Encode message buffer object
`elemName` - Element name

decodeXER

```
public void decodeXER(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

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

encode

```
public void encode(Asn1XmlEncoder buffer,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an ASN.1 Universal String value with element and attribute 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 prefix value

encodeData

```
public void encodeData(Asn1XmlXerEncoder buffer)
  throws java.io.IOException,
         Asn1Exception
```

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

decodeXML

```
public void decodeXML(java.lang.String buffer,
         java.lang.String attrs)
  throws Asn1Exception
```

This method decodes an ASN.1 Universal String value using the XML schema encoding rules.

Parameters:

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

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 restricted character string from JSON.

Parameters:

`buffer`

(continued from last page)

Throws:
`java.io.IOException`

encode

```
public void encode(Asn1JsonOutputStream out)
  throws java.io.IOException
```

Encode the value of this object as a JSON string.

Parameters:
`out`

equals

```
public boolean equals(java.lang.Object cs_)
```

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

Parameters:
`cs_` - String value

hashCode

```
public int hashCode()
```

This method returns the hashcode for this object.

getLength

```
public int getLength()
  throws Asn1InvalidLengthException
```

This method will return the length of the character string in characters.

Returns:
Number of characters.

toString

```
public java.lang.String toString()
```

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

Returns:
Stringified representation of the value

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:
`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

(continued from last page)

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
          Asn1CharSet charSet)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:

`out` - PER Output Stream object
`charSet` - Object representing permitted alphabet constraint character set (optional)

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 **value**member variable.

Parameters:

`out` - 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

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

encode

```
protected void encode(Asn1PerOutputStream out,
          int nchars,
          int offset,
          int abpc,
          int ubpc,
          Asn1CharSet charSet)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes the contents of a **UniversalString**type. This version assumes a permitted alphabet constraint was specified.

Parameters:

`out` - 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)

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
protected void encode(Asn1PerOutputStream out,
          int abpc,
          int ubpc,
          Asn1CharSet charSet)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**member variable.

Parameters:

`out` - 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)

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
protected void encode(Asn1PerOutputStream out,
          int abpc,
          int ubpc,
          Asn1CharSet charSet,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 **value**member variable.

Parameters:

(continued from last page)

`out` - 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

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

validate

`public boolean` **`validate`**`(Asn1CharSet charSet)`

This method will attempt to validate a string against its internal character set.

Returns:

True or False.

com.objsys.asn1j.runtime
Class Asn1UnknownIEI

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1UnknownIEI
```

public class **Asn1UnknownIEI**
extends Asn1Exception

This exception is thrown when a 3GPP message has an unknown IEI which requires comprehension.

Constructor Summary

public	`Asn1UnknownIEI(int iei)` This constructor creates an exception object with a default textual message.

Method Summary

int	`getIEI()`

Methods inherited from class `java.lang.Throwable`
`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1UnknownIEI

`public` **`Asn1UnknownIEI`**`(int iei)`

This constructor creates an exception object with a default textual message.

Methods

(continued from last page)

getIEI

```
public final int getIEI()
```

com.objsys.asn1j.runtime
Class Asn1UTCTime

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1AbstractTime
               ┗-com.objsys.asn1j.runtime.Asn1UTCTime
```

public class **Asn1UTCTime**
extends Asn1AbstractTime

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 23).

Fields inherited from class com.objsys.asn1j.runtime.Asn1AbstractTime
Apr, April, Aug, August, day, Dec, December, derRules, diffHour, diffMin, Feb, February, hour, Jan, January, Jul, July, Jun, June, Mar, March, May, minute, month, Nov, November, Oct, October, parsed, secFraction, second, Sep, September, utcFlag, year

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1UTCTime() The default constructor creates an empty time string object.

public	`Asn1UTCTime(boolean useDerRules)` This constructor creates an empty time string object and allows DER encoding rules to be specified.
public	`Asn1UTCTime(java.lang.String data)` This version of the constructor can be used to set the string **value**member variable to the given time string.
public	`Asn1UTCTime(java.lang.String data,boolean useDerRules)` This version of the constructor can be used to set the string **value**member variable to the given time string and specify DER encoding rules be used to construct the string.

Method Summary

void	`clear()` Clears out time string.
boolean	`compileString()` Compiles new time string according to X.680 (clause 42) and ISO 8601.
void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
boolean	`equals(java.lang.String s)` This method compares this object with a String value.
java.lang.String	`getFraction()` This method always returns zero for this class.
void	`init()`
void	`parseString(java.lang.String string)` This method parses passed time string.
void	`setFraction(java.lang.String fraction)` This method is not supported for UTC time.
void	`setTime(java.util.Calendar time)` This method converts the java.util.Calendar value to time string.
void	`setYear(int year)` This method sets the year component of the time value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1AbstractTime`

charAt, clear, compareTo, compileString, decode, decode, decodeXML, encode, encode, encode, encode, encode, encodeXER, encodeXMLData, equals, equals, getDay, getDiff, getDiffHour, getDiffMinute, getFraction, getHour, getMinute, getMonth, getSecond, getTime, getUTC, getYear, init, parseInt, parseString, parseXmlString, putInteger, putInteger, safeParseString, setDay, setDER, setDiff, setDiff, setDiffHour, setFraction, setHour, setMinute, setMonth, setSecond, setTime, setUTC, setYear

Methods inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 23).

Constructors

Asn1UTCTime

`public` **`Asn1UTCTime`**`()`

The default constructor creates an empty time string object.

Asn1UTCTime

`public` **`Asn1UTCTime`**`(boolean 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.

Asn1UTCTime

```
public Asn1UTCTime(java.lang.String data)
```

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

Parameters:

`data -` Character string containing UTC time value to be set. The format of the string is YYYYMMDDHH[MM](Z|(+|-)HH(MM))

Asn1UTCTime

```
public Asn1UTCTime(java.lang.String data,
                   boolean useDerRules)
```

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

Parameters:

`data -` Character string containing UTC time value to be set. The format of the string is YYYYMMDDHH[MM](Z|(+|-)HH(MM))
`useDerRules -` 'true' if time string should be encoded with DER/PER.

Methods

init

```
protected void init()
```

equals

```
public boolean equals(java.lang.String s)
```

This method compares this object with a String value. Strictly speaking, the contract of equals stipulates identity: a.equals(b) iff b.equals(a). However, it is never the case that a String will equal an Asn1Time object. In this case, then, equals should be taken to mean that the time represented by the string passed in is the same time as the object to which it is compared.

Parameters:

`s`

Returns:

True, if the input string represents the same time as this object.

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
                   boolean explicit,
                   int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit -` Flag indicating element is explicitly tagged
`implicitLength -` Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

clear

```
public void clear()
```

Clears out time string.

setYear

```
public void setYear(int year)
  throws Asn1Exception
```

This method 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 'getYear' method returns year in full 4 digits format, independently of format of 'year' parameter for 'setYear' method.

Parameters:

`year` - Year component (full 4 digits).

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setFraction

```
public void setFraction(java.lang.String fraction)
  throws Asn1Exception
```

This method is not supported for UTC time.

Parameters:

`fraction` - Any integer value.

Throws:

`Asn1Exception` - Always is thrown.

getFraction

```
public java.lang.String getFraction()
  throws Asn1Exception
```

This method always returns zero for this class.

Returns:

Zero.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

setTime

```
public void setTime(java.util.Calendar time)
  throws Asn1Exception
```

This method converts the java.util.Calendar value to time string. Note that the action of this method may be different for different inherited Asn1Time classes.

Parameters:

`time` - The calendar time value.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

parseString

```
public void parseString(java.lang.String string)
  throws Asn1Exception
```

This method parses passed time string.

Parameters:

`string` - Character string containing UTC time value to be parsed. The format of the string is YYYYMMDDHH[MM](Z|(+|-)HH(MM))

Throws:

`Asn1Exception` - Thrown if format of string is not correct.

compileString

```
protected boolean compileString()
  throws Asn1Exception
```

Compiles new time string according to X.680 (clause 42) and ISO 8601.

Returns:

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

Throws:

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1UTF8String

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1UTF8String
```

Direct Known Subclasses:
Asn1XmlAnyElem, Asn1Real10

---

public class **Asn1UTF8String**
extends Asn1CharString

This is a container class for holding the components of an ASN.1 UTF-8 string value.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 12).

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1UTF8String() The default constructor creates an empty time string object.
public	Asn1UTF8String(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string value.

Method Summary

static java.lang.String	decode(Asn1BerDecodeBuffer buffer,Asn1Tag explicitTag,int implicitLength) This method decodes an ASN.1 UTF-8 string value including the UNIVERSAL tag value and length if explicit tagging is specified.

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 UTF-8 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1OerDecodeBuffer buffer)` This method decodes an ASN.1 UTF8String string value that was encoded according to OER.
void	`decode(Asn1OerDecodeBuffer buffer,int length)` This method decodes the content of an ASN.1 UTF8String string value that was encoded according to OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 UTF-8 string value using the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This method decodes a sized ASN.1 UTF-8 string value using the packed encoding rules (PER).
static java.lang.String	`decodeUTF8(Asn1OerDecodeBuffer buffer)` This method decodes the content of an ASN.1 UTF8String string value that was encoded according to OER.
static java.lang.String	`decodeUTF8(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 UTF-8 string value using the packed encoding rules (PER).
static int	`encode(Asn1BerEncodeBuffer buffer,Asn1Tag explicitTag,java.lang.String value)` This method encodes a ASN.1 UTF8String.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
static void	`encode(Asn1BerOutputStream out,Asn1Tag explicitTag,java.lang.String value)` This method encodes the given string using the BER encoding rules for UTF8String, including the given tag, if provided.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1OerEncodeBuffer buffer)` This method encodes a ASN.1 UTF8String according to OER.
static void	`encode(Asn1OerEncodeBuffer buffer,java.lang.String value)` This method encodes an ASN.1 UTF8String according to OER.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an unconstrained ASN.1 UTF-8 string value using the packed encoding rules (PER).

void	encode(Asn1PerEncodeBuffer buffer,long lower,long upper) This method encodes a size-constrained ASN.1 UTF-8 string value using the packed encoding rules (PER).
static void	encode(Asn1PerEncodeBuffer buffer,java.lang.String value) This method encodes a string using the packed encoding rules (PER) specified for ASN.1 UTF8String.
void	encode(Asn1PerOutputStream out) This method encodes an unconstrained ASN.1 UTF-8 string value using the packed encoding rules (PER).
void	encode(Asn1PerOutputStream out,long lower,long upper) This method encodes a size-constrained ASN.1 UTF-8 string value using the packed encoding rules (PER).
void	setAnyAttribute(java.lang.String qname,java.lang.String val) This method will set the anyAttribute type value for given qname and value of XML attribute

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

TAG

public static final com.objsys.asn1j.runtime.Asn1Tag **TAG**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 12).

Constructors

Asn1UTF8String

public **Asn1UTF8String**()

The default constructor creates an empty time string object.

Asn1UTF8String

```
public Asn1UTF8String(java.lang.String data)
```

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

Methods

decode

```
public static java.lang.String decode(Asn1BerDecodeBuffer buffer,
         Asn1Tag explicitTag,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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. BER encodes UTF8String, OID-IRI, and RELATIVE-OID-IRI in essentially the same way; this permits sharing of the implementation.

Parameters:

`buffer` - Decode message buffer object

`explicitTag` - Tag to explicitly match, or null if none

`implicitLength` - Length of contents if explicitTag is not given

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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

`explicit` - Flag indicating element is explicitly tagged

`implicitLength` - Length of contents if implicit

encode

```
public static int encode(Asn1BerEncodeBuffer buffer,
         Asn1Tag explicitTag,
         java.lang.String value)
  throws Asn1Exception
```

This method encodes a ASN.1 UTF8String. Nearly the same encoding is shared by OID-IRI and RELATIVE-OID_IRI; this method facilitates sharing the implementation. The UNIVERSAL tag value and length is also encoded if explicit tagging is specified.

Parameters:

`buffer` - Encode message buffer object

`explicitTag` - Tag to explicitly encode, or null for none.

`value` - The value to encode. For OID-IRI and RELATIVE-OID-IRI, whitespace should already have been removed.

Returns:

Length in octets of encoded component

(continued from last page)

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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

`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

---

decodeUTF8

```
public static java.lang.String decodeUTF8(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an ASN.1 UTF-8 string value using the packed encoding rules (PER).

Parameters:

`buffer` - Decode message buffer object

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         long lower,
         long upper)
  throws Asn1Exception,
         java.io.IOException
```

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

---

encode

```
public static void encode(Asn1PerEncodeBuffer buffer,
         java.lang.String value)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes a string using the packed encoding rules (PER) specified for ASN.1 UTF8String. These rules are essentially shared with OID-IRI and RELATIVE-OID-IRI.

(continued from last page)

Parameters:

`buffer` - Encode message buffer object

`value` - The value to be encoded. In the case of OID-IRI and RELATIVE-OID-IRI, should not contain whitespace.

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variable within this class.

Parameters:

`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' 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

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes an ASN.1 UTF8String string value that was encoded according to OER.

Parameters:

`buffer` - Decode message buffer object

decode

```
protected void decode(Asn1OerDecodeBuffer buffer,
          int length)
  throws java.io.IOException
```

This method decodes the content of an ASN.1 UTF8String string value that was encoded according to OER. The length is not decoded but is taken to be as given.

Parameters:

`buffer` - Decode message buffer object

decodeUTF8

```
public static java.lang.String decodeUTF8(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

This method decodes the content of an ASN.1 UTF8String string value that was encoded according to OER.

Parameters:

`buffer` - Decode message buffer object

(continued from last page)

Returns:
The decoded string.

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

This method encodes a ASN.1 UTF8String according to OER.

Parameters:
`buffer` - Encode message buffer object

encode

```
public static void encode(Asn1OerEncodeBuffer buffer,
         java.lang.String value)
  throws java.io.IOException
```

This method encodes an ASN.1 UTF8String according to OER.

Parameters:
`buffer` - Encode message buffer object
`value` - The value to be encoded.

setAnyAttribute

```
public void setAnyAttribute(java.lang.String qname,
         java.lang.String val)
```

This method will set the anyAttribute type value for given qname and value of XML attribute

Parameters:
`qname` - The qualified (prefixed) name
`val` - The attribute value

encode

```
public static void encode(Asn1BerOutputStream out,
         Asn1Tag explicitTag,
         java.lang.String value)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes the given string using the BER encoding rules for UTF8String, including the given tag, if provided.

Parameters:
`out` - BER Output Stream object
`explicitTag` - tag to encode or null for none
`value` - The value to encode. For OID-IRI and RELATIVE-OID-IRI, should not contain whitespace.

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variable within this class.

Parameters:

`out` - PER Output Stream object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 'value' public member variable within this class.

Parameters:

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

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1Util

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Util
```

public class **Asn1Util**
extends java.lang.Object

This class contains some general purpose static utility functions.

Constructor Summary

public	Asn1Util()

Method Summary

static java.lang.String	BCDToString(byte[] bcd) Translates a BCD string to an ASCII string.
static void	closeRuntime() Signal that you are finished with the ASN1C runtime and release internal resources.
static byte[]	decodeBase64Array(char[] srcArray) Translates the specified Base64 array into byte array.
static java.lang.String	encodeBase64Array(byte[] srcArray) Translates the specified byte array to Base64 string.
static java.lang.String	firstSortedElement(java.util.Vector v) Returns the element of a Vector (that is assumed to contain strings) that is lexicographically the lowest element in the Vector.
static int	getBytesCount(long val) Calculate the count of bytes necessary to represent a signed long value.
static java.util.Vector	getKeys(java.util.Hashtable h) Returns a Vector containing the keys (assumed to be strings) of a passed Hashtable.
static int	getUlongBytesCount(long val) Calculate the count of bytes necessary to represent an unsigned long value.
static int	hexToByte(char c) Return the value of a single hexadecimal character.
static boolean	isLetter(char c) Determine if a specified character is a letter.

static boolean	isLimited() Indicates whether the run-time is limited or unlimited.
static boolean	isSpaceChar(char ch) Determine if a specified character is a space.
static boolean	isWhitespace(char ch) Determine if a specified character is whitespace.
static double	log2(double val) Calculate the log base 2 of value
static double	pow(int base,int exponent) Raises an integer base to an integer power and returns the result as a double.
static int	rotateLeft(int value,int bitCnt) Return the result of rotating the bits in value left by bitCnt positions.
static java.lang.String[]	splitOnWhitespace(java.lang.String delimitedString) Split a string around whitespace characters.
static byte[]	stringToBCD(java.lang.String str) Translates an ASCII string to a BCD string.
static byte[]	stringToTBCD(java.lang.String str) Translates an ASCII String to a TBCD String.
static java.lang.String	stripWhitespace(java.lang.String value) Return the given string with all whitespace characters removed.
static char	tbcdBinToChar(byte tbcdDigit) This function converts a TBCD binary character into its ASCII equivalent.
static byte	tbcdCharToBin(char digit) This function converts a TBCD character ('0'-'9',"*#abc") into its binary equivalent.
static java.lang.String	TBCDToString(byte[] bcd) Translates a TBCD string to an ASCII string.
static char	toHexChar(int nibble) Convert a value from 0x0 to 0xF to the corresponding hex char.
static java.lang.String	toHexString(byte b) Convert a byte value to a hex string.
static java.lang.String	toHexString(byte[] b,int offset,int nbytes) Convert a string of bytes into a hex string.
static java.lang.StringBuffer	toHexString(byte b,java.lang.StringBuffer hexbuf) Convert a byte value to a hex string.

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1Util

public **Asn1Util**()

Methods

closeRuntime

public static void **closeRuntime**()

Signal that you are finished with the ASN1C runtime and release internal resources. Do not use the runtime classes, on any thread, after calling this. This may be called from any thread.

stripWhitespace

public static java.lang.String **stripWhitespace**(java.lang.String value)

Return the given string with all whitespace characters removed. Here, "whitespace characters" means those characters defined by X.680 12.1.6 as whitespace: HT, LF, VT, FF, CR, SPACE.

Parameters:

value

Returns:

value, with all whitespace removed; if the input string has no whitespace, it is returned itself.

hexToByte

public static int **hexToByte**(char c)

Return the value of a single hexadecimal character.

Parameters:

c - The hex character (0-9,A-F,a-f)

Returns:

Value of hex character, 0-15.

Throws:

Asn1Exception - if charater is not a hex character.

toHexString

public static java.lang.String **toHexString**(byte b)

Convert a byte value to a hex string. Unlike the Java 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.

(continued from last page)

toHexChar

```
public static char toHexChar(int nibble)
```

Convert a value from 0x0 to 0xF to the corresponding hex char. Lowercase letters are used for the hex characters.

---

toHexString

```
public static java.lang.StringBuffer toHexString(byte b,
         java.lang.StringBuffer hexbuf)
```

Convert a byte value to a hex string. Unlike the Java 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 or equal to 0xf In other words, a character string of length 2 is always appended to hexbuf.

---

toHexString

```
public static java.lang.String toHexString(byte[] b,
         int offset,
         int nbytes)
```

Convert a string of bytes into a hex string.

---

getBytesCount

```
public static int getBytesCount(long val)
```

Calculate the count of bytes necessary to represent a signed long value.

Parameters:

`val` - Integer value in which to count bytes.

Returns:

Bit count.

---

log2

```
public static double log2(double val)
```

Calculate the log base 2 of value

Parameters:

`val` - double value for which to count log(base2) val.

Returns:

result value.

---

getUlongBytesCount

```
public static int getUlongBytesCount(long val)
```

Calculate the count of bytes necessary to represent an unsigned long value.

Parameters:

`val` - Integer value in which to count bytes.

Returns:

Bit count.

---

encodeBase64Array

```
public static java.lang.String encodeBase64Array(byte[] srcArray)
```

(continued from last page)

Translates the specified byte array to Base64 string.

Parameters:

`srcArray -` byte array to be translated

Returns:

Base64 string

---

decodeBase64Array

```
public static byte[] decodeBase64Array(char[] srcArray)
```

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

---

stringToBCD

```
public static byte[] stringToBCD(java.lang.String str)
  throws Asn1ValueParseException
```

Translates an ASCII string to a BCD string. The ASCII string must contain only the following characters: [0..9A-Ea-e]. 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.

Throws:

`Asn1ValueParseException -` If invalid characters are in the source string.

BCDToString

```
public static java.lang.String BCDToString(byte[] bcd)
```

Translates a BCD string to an ASCII string. The returned string will consist of characters in [0..9] and [A-E]. All half-bytes in the input shall be less than 0xF, except that in the final byte, the low half-byte may be 0xF, to act as filler for a string of odd length.

Parameters:

`bcd -` the source BCD string

Returns:

the ASCII string.

---

stringToTBCD

```
public static byte[] stringToTBCD(java.lang.String str)
  throws Asn1ValueParseException
```

Translates an ASCII String to a TBCD String. The TBCD encoding format is as described in 3GPP 29.002: - the characters used are [0-9*#abc]. - each nibble represents a digit, and the upper nibble represents the digit which follows the digit represented in the lower nibble. - 0xF is used as filler in the final high nibble when the input string has and odd length. The ASCII string "12345" would be encoded 0x2143f5.

Parameters:

(continued from last page)

`str` - the source ASCII string

Returns:
the BCD string as a byte array.

Throws:
`Asn1ValueParseException` - If invalid characters are in the source string.

TBCDToString

`public static java.lang.String` **`TBCDToString`**`(byte[] bcd)`

Translates a TBCD string to an ASCII string. Refer to the description of TBCD in stringToTBCD.

Parameters:
`bcd` - the source TBCD string

Returns:
the ASCII string.

---

tbcdBinToChar

`public static char` **`tbcdBinToChar`**`(byte tbcdDigit)`

This function converts a TBCD binary character into its ASCII equivalent.

Parameters:
`tbcdDigit` - TBCD digit

Returns:
the converted character

---

tbcdCharToBin

`public static byte` **`tbcdCharToBin`**`(char digit)`

This function converts a TBCD character ('0'-'9',"*#abc") into its binary equivalent.

Parameters:
`digit` - TBCD digit character ('0'-'9',"*#abc")

Returns:
binary representation of given char

---

isLimited

`public static boolean` **`isLimited`**`()`

Indicates whether the run-time is limited or unlimited.

Returns:
true if limited, false if unlimited

---

pow

`public static double` **`pow`**`(int base, int exponent)`

Raises an integer base to an integer power and returns the result as a double.

Parameters:
`base` - the base to be raised to the power.

(continued from last page)

`exponent` - the power to which base is to be raised.

Returns:

base raised to exponent as a double.

---

rotateLeft

```
public static int rotateLeft(int value,
                             int bitCnt)
```

Return the result of rotating the bits in value left by bitCnt positions.

Parameters:

`value`
`bitCnt`

Returns:

---

isWhitespace

```
public static boolean isWhitespace(char ch)
```

Determine if a specified character is whitespace.

Parameters:

`ch` - the character to test.

Returns:

true if the character is whitespace, false if it isn't.

---

isSpaceChar

```
public static boolean isSpaceChar(char ch)
```

Determine if a specified character is a space.

Parameters:

`ch` - the character to test.

Returns:

true if the character is a space, false if it isn't.

---

firstSortedElement

```
public static java.lang.String firstSortedElement(java.util.Vector v)
```

Returns the element of a Vector (that is assumed to contain strings) that is lexicographically the lowest element in the Vector.

Parameters:

`v` - the Vector to be examined.

Returns:

the lowest value in the Vector as a String.

---

getKeys

```
public static java.util.Vector getKeys(java.util.Hashtable h)
```

Returns a Vector containing the keys (assumed to be strings) of a passed Hashtable.

(continued from last page)

Parameters:

`h` - the Hashtable to be examined.

Returns:

a Vector containing the keys.

isLetter

`public static boolean` **`isLetter`**`(char c)`

Determine if a specified character is a letter.

Parameters:

`c` - The character to examine.

Returns:

true if the character is a letter, false otherwise.

splitOnWhitespace

`public static java.lang.String[]` **`splitOnWhitespace`**`(java.lang.String delimitedString)`

Split a string around whitespace characters.

Parameters:

`delimitedString` - the whitespace-delimited string to parse.

Returns:

an array of Strings with each element being a piece of the original string.

com.objsys.asn1j.runtime
Class Asn1Value

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1Value
```

public class **Asn1Value**
extends java.lang.Object

This class provides methods for parsing and formatting text in ASN.1 value notation.

Constructor Summary

public	Asn1Value()

Method Summary

static byte[]	parseString(java.lang.String value) This overloaded version of the parseString method sets the numbits holder value to null.
static byte[]	parseString(java.lang.String value,IntHolder numbits) 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.
static boolean	stringEqualsBytes(java.lang.String value,byte[] data) This static method parses the given ASN.1 value text (either a binary or hex data string) and compares it with the given byte array.
static boolean	stringEqualsBytes(java.lang.String value,byte[] data,int nbits) This static method parses the given ASN.1 value text (either a binary or hex data string) and compares it with the given byte array.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1Value

public **Asn1Value**()

Methods

(continued from last page)

stringEqualsBytes

```
public static boolean stringEqualsBytes(java.lang.String value,
         byte[] data)
```

This static method parses the given ASN.1 value text (either a binary or hex data string) and compares it with the given byte array. 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

`data` - Array of bytes to compare string with.

---

stringEqualsBytes

```
public static boolean stringEqualsBytes(java.lang.String value,
         byte[] data,
         int nbits)
  throws Asn1ValueParseException
```

This static method parses the given ASN.1 value text (either a binary or hex data string) and compares it with the given byte array. This version of the method allows you to specify a number of bits to compare, which is useful when dealing with bit strings, as opposed to octet strings. 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

`data` - Array of bytes to compare string with.

`nbits` - Number of bits in data to compare.

---

parseString

```
public static byte[] parseString(java.lang.String value,
         IntHolder numbits)
  throws Asn1ValueParseException
```

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:

`value` - The ASN.1 value specification text

`numbits` - Holder to receive number of bits in string

---

parseString

```
public static byte[] parseString(java.lang.String value)
  throws Asn1ValueParseException
```

This overloaded version of the parseString method sets the numbits holder value to null.

Parameters:

`value` - The ASN.1 value specification text

com.objsys.asn1j.runtime
Class Asn1ValueParseException

```
java.lang.Object
   |-java.lang.Throwable
      |-java.lang.Exception
         |-java.lang.RuntimeException
            |-com.objsys.asn1j.runtime.Asn1Exception
               |-com.objsys.asn1j.runtime.Asn1ValueParseException
```

public class **Asn1ValueParseException**
extends Asn1Exception

This class defines the 'ASN.1 value parse' exception that is thrown when a string containing an ASN.1 value cannot be parsed.

Constructor Summary

public	`Asn1ValueParseException(java.lang.String text)` This constructor creates an exception object with a textual message describing the expected and parsed tag values.
public	`Asn1ValueParseException(java.lang.String text,int offset)` This constructor creates an exception object with a textual message describing the expected and parsed tag values.

Methods inherited from class `java.lang.Throwable`
`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1ValueParseException

`public` **`Asn1ValueParseException`**`(java.lang.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

(continued from last page)

Asn1ValueParseException

```
public Asn1ValueParseException(java.lang.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` - Byte offset to error location in string

com.objsys.asn1j.runtime
Class Asn1ValueRange

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1ValueRange
```

public class **Asn1ValueRange**
extends java.lang.Object

This class models an integer value range. These value ranges may be created by users for queries made to generated SEQUENCEor SETtypes.

Field Summary

long	`max` The maximum value taken on by this range.
long	`min` The minimum value taken on by this range.

Constructor Summary

public	`Asn1ValueRange()` This constructor sets the minimum and maximum values to the limits set in java.lang.Long.
public	`Asn1ValueRange(long min,long max)` This constructor takes a minimum and maximum value to set the range.
public	`Asn1ValueRange(java.lang.Object min,long max)` This constructor ignores the min value and sets the maximal value for the range.
public	`Asn1ValueRange(long min,java.lang.Object max)` This constructor ignores the max value and sets the minimal value for the range.

Method Summary

long	`getMaxValue()` This method returns the maximum value.
long	`getMinValue()` This method returns the minimum value.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

min

`public long min`

The minimum value taken on by this range.

max

`public long max`

The maximum value taken on by this range.

Constructors

Asn1ValueRange

`public Asn1ValueRange()`

This constructor sets the minimum and maximum values to the limits set in java.lang.Long.

Asn1ValueRange

```
public Asn1ValueRange(long min,
                      long max)
```

This constructor takes a minimum and maximum value to set the range.

Parameters:

`min` - The minimum value.
`max` - The maximum value.

Asn1ValueRange

```
public Asn1ValueRange(java.lang.Object min,
                      long max)
```

This constructor ignores the min value and sets the maximal value for the range. In order to call it properly, pass a nullvalue for the minimum.

Asn1ValueRange

```
public Asn1ValueRange(long min,
                      java.lang.Object max)
```

This constructor ignores the max value and sets the minimal value for the range. In order to call it properly, pass a nullvalue for the maximum.

Methods

getMinValue

`public long getMinValue()`

This method returns the minimum value.

(continued from last page)

getMaxValue

`public long` **`getMaxValue`**`()`

This method returns the maximum value.

com.objsys.asn1j.runtime
Class Asn1VarWidthCharString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1VarWidthCharString
```

Direct Known Subclasses:
Asn1VideotexString, Asn1T61String, Asn1ObjectDescriptor, Asn1GraphicString, Asn1GeneralString

---

public abstract class **Asn1VarWidthCharString**
extends Asn1CharString

This is an abstract base class for holding the ASN.1 variable width character string types (GraphicString, GeneralString, TeletexString, T61String, VideotexString, ObjectDescriptor).

Field Summary

static final int	BITSPERCHAR_A The BITSPERCHAR_Aconstant specifies the number of bits per character for PER (aligned). Value: **8**
static final int	BITSPERCHAR_U The BITSPERCHAR_Uconstant specifies the number of bits per character for PER (unaligned). Value: **8**

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString

mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

protected	Asn1VarWidthCharString(short typeCode) The default constructor creates an empty string object.
protected	Asn1VarWidthCharString(java.lang.String data,short typeCode) This version of the constructor can be used to set the string valuemember variable to the given string.

Method Summary

void	`decode(Asn1OerDecodeBuffer buffer)` Decode the value in accordance with OER.
void	`decode(Asn1PerDecodeBuffer buffer)` This method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`decode(Asn1PerDecodeBuffer buffer,long lower,long upper)` This overloaded version of the decode method decodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1OerEncodeBuffer buffer)` Encode the value in accordance with OER.
void	`encode(Asn1PerEncodeBuffer buffer)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This overloaded version of the encode method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an ASN.1 character string value in accordance with the packed encoding rules (PER) directly into the stream.
void	`encode(Asn1PerOutputStream out,long lower,long upper)` 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).

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

(continued from last page)

BITSPERCHAR_A

```
public static final int BITSPERCHAR_A
```

The BITSPERCHAR_Aconstant specifies the number of bits per character for PER (aligned).

BITSPERCHAR_U

```
public static final int BITSPERCHAR_U
```

The BITSPERCHAR_Uconstant specifies the number of bits per character for PER (unaligned).

Constructors

Asn1VarWidthCharString

```
protected Asn1VarWidthCharString(short typeCode)
```

The default constructor creates an empty string object.

Parameters:

`typeCode` - Universal ID code for ASN.1 character string

Asn1VarWidthCharString

```
protected Asn1VarWidthCharString(java.lang.String data,
                                 short typeCode)
```

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

Parameters:

`data` - Character string
`typeCode` - Universal ID code for ASN.1 character string

Methods

decode

```
public void decode(Asn1PerDecodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`buffer` - Decode message buffer object

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

(continued from last page)

`buffer` - Decode message buffer object
`lower` - Effective size constraint lower bound
`upper` - Effective size constraint upper bound

encode

```
public void encode(Asn1PerEncodeBuffer buffer)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:
`buffer` - Encode message buffer object

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
         java.io.IOException
```

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 `value`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

decode

```
public void decode(Asn1OerDecodeBuffer buffer)
  throws java.io.IOException
```

Decode the value in accordance with OER. Subclasses may override this to add constraint checks after invoking this method to decode the string.

Parameters:
`buffer`

encode

```
public void encode(Asn1OerEncodeBuffer buffer)
  throws java.io.IOException
```

Encode the value in accordance with OER. We're assuming each char is encoded in a single byte of the same value as the char. Subclasses may override this (e.g. to add constraint checks) and invoke this method to perform the encoding.

Parameters:
`buffer`

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
         java.io.IOException
```

(continued from last page)

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object

Throws:

`java.io.IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encode

```
public void encode(Asn1PerOutputStream out,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

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 `value`member variable in the **Asn1CharString**base class.

Parameters:

`out` - PER Encode message stream object
`lower` - Effective size constraint lower bound
`upper` - Effective size constraint upper bound

Throws:

`java.io.IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1VideotexString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1VarWidthCharString
            ┗-com.objsys.asn1j.runtime.Asn1VideotexString
```

public class **Asn1VideotexString**
extends Asn1VarWidthCharString

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

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 21).

Fields inherited from class com.objsys.asn1j.runtime.Asn1VarWidthCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1VideotexString() The default constructor creates an empty string object.
public	Asn1VideotexString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 string value including the UNIVERSAL tag value and length if explicit tagging is specified.
void	`decode(Asn1JsonDecodeBuffer buffer)` Decode ASN.1 VideotexString from JSON.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.
void	`encode(Asn1JsonOutputStream outstream)` Encode this VideotexString to JSON.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1VarWidthCharString`
`decode, decode, decode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`
`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`
`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 21).

Constructors

(continued from last page)

Asn1VideotexString

```
public Asn1VideotexString()
```

The default constructor creates an empty string object.

Asn1VideotexString

```
public Asn1VideotexString(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

Throws:

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

(continued from last page)

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

---

decode

```
public void decode(Asn1JsonDecodeBuffer buffer)
  throws java.io.IOException
```

Decode ASN.1 VideotexString from JSON.

Parameters:

`buffer` - The buffer to decode from.

Throws:

`java.io.IOException` - for I/O exception

---

encode

```
public void encode(Asn1JsonOutputStream outstream)
  throws java.io.IOException
```

Encode this VideotexString to JSON.

Parameters:

`outstream` - The stream to encode to.

Throws:

`java.io.IOException` - for I/O exception

com.objsys.asn1j.runtime
Class Asn1VisibleString

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn18BitCharString
            ┗-com.objsys.asn1j.runtime.Asn1VisibleString
```

public class **Asn1VisibleString**
extends Asn18BitCharString

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

Field Summary

static final Asn1CharSet	CHARSET
static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 26).

Fields inherited from class com.objsys.asn1j.runtime.Asn18BitCharString
BITSPERCHAR_A, BITSPERCHAR_U

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1VisibleString() The default constructor creates an empty string object.
public	Asn1VisibleString(java.lang.String data) This version of the constructor can be used to set the string **value**member variable to the given string.

Method Summary

void	`decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength)` This method decodes an ASN.1 Visible string value including the UNIVERSAL tag value and length if explicit tagging is specified.
int	`encode(Asn1BerEncodeBuffer buffer,boolean explicit)` This method encodes an ASN.1 Visible string type.
void	`encode(Asn1BerOutputStream out,boolean explicit)` 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.

Methods inherited from class `com.objsys.asn1j.runtime.Asn18BitCharString`

`decode, decode, decode, decode, decode, encode, encode, encode, encode, encode, encode, encode, encode`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1CharString`

`decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

CHARSET

`public static final com.objsys.asn1j.runtime.Asn1CharSet` **`CHARSET`**

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 26).

Constructors

(continued from last page)

Asn1VisibleString

```
public Asn1VisibleString()
```

The default constructor creates an empty string object.

Asn1VisibleString

```
public Asn1VisibleString(java.lang.String data)
```

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

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

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
`explicit` - Flag indicating element is explicitly tagged
`implicitLength` - Length of contents if implicit

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

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
`explicit` - Flag indicating explicit tagging should be done

Returns:

Length in octets of encoded component

encode

```
public void encode(Asn1BerOutputStream out,
         boolean explicit)
  throws Asn1Exception,
         java.io.IOException
```

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).

Parameters:

`out` - BER Output Stream object
`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1X694OrderElement

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1X694OrderElement
```

public class **Asn1X694OrderElement**
extends Asn1Type

This is a helper class for X.694 order element.

Field Summary

static final Asn1Tag	TAG The **TAG**constant describes the universal tag for this data type (UNIVERSAL 10).
int	value This public member variable is where the _order element value is stored.

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type

BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1X694OrderElement() The default constructor sets the _order value to zero.
public	Asn1X694OrderElement(int value_) This constructor creates an Asn1X694OrderElement object from given integer array.

Method Summary

void	decode(Asn1BerDecodeBuffer buffer,boolean explicit,int implicitLength) This method decodes an X.694 order element value including the tag value and length if explicit tagging is specified.
void	decode(Asn1PerDecodeBuffer buffer,int lower,int upper) This method decodes an unconstrained X.694 order element value using the Packed Encoding Rules (PER).
int	encode(Asn1BerEncodeBuffer buffer,boolean explicit) This method encodes an X.694 order element value including the tag value and length if explicit tagging is specified.

void	`encode(Asn1BerOutputStream out,boolean explicit)` This method encodes and writes to the stream an X.694 order element value including the tag value and length if explicit tagging is specified.
void	`encode(Asn1PerEncodeBuffer buffer,long lower,long upper)` This method encodes an unconstrained X.694 order element value using the Packed Encoding Rules (PER).
void	`encode(Asn1PerOutputStream out)` This method encodes an unconstrained X.694 order element value using the Packed Encoding Rules (PER).
boolean	`equals(java.lang.Object o)` This method compares this X.694 order value to the given value for equality.
int	`hashCode()` This method will return the hash code for this X.694 order value.
java.lang.String	`toString()` This method will return a string representation of the order integer value.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1Type`

`_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

TAG

`public static final com.objsys.asn1j.runtime.Asn1Tag` **`TAG`**

The **TAG**constant describes the universal tag for this data type (UNIVERSAL 10).

value

`public transient int` **`value`**

This public member variable is where the _order element 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.

Constructors

(continued from last page)

Asn1X694OrderElement

`public` **`Asn1X694OrderElement`**`()`

The default constructor sets the _order value to zero.

Asn1X694OrderElement

`public` **`Asn1X694OrderElement`**`(int value_)`

This constructor creates an Asn1X694OrderElement object from given integer array.

Parameters:

`value_` - Integer conatins element order value

Methods

decode

```
public void decode(Asn1BerDecodeBuffer buffer,
         boolean explicit,
         int implicitLength)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an X.694 order element value including the tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters:

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

---

encode

```
public int encode(Asn1BerEncodeBuffer buffer,
         boolean explicit)
  throws Asn1Exception
```

This method encodes an X.694 order element value including the tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters:

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

Returns:

Length of component or negative status value

---

decode

```
public void decode(Asn1PerDecodeBuffer buffer,
         int lower,
         int upper)
  throws Asn1Exception,
         java.io.IOException
```

This method decodes an unconstrained X.694 order element 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 'value' in this object.

Parameters:

`buffer` - PER Decode message buffer object

encode

```
public void encode(Asn1PerEncodeBuffer buffer,
          long lower,
          long upper)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an unconstrained X.694 order element value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters:

`buffer` - PER Encode message buffer object

equals

```
public boolean equals(java.lang.Object o)
```

This method compares this X.694 order value to the given value for equality.

Parameters:

`o` - Asn1X694OrderElement object containing value to compare

hashCode

```
public int hashCode()
```

This method will return the hash code for this X.694 order value.

toString

```
public java.lang.String toString()
```

This method will return a string representation of the order integer value.

Returns:

Stringified representation of the value

encode

```
public void encode(Asn1BerOutputStream out,
          boolean explicit)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes and writes to the stream an X.694 order element value including the tag value and length if explicit tagging is specified. This overloaded version uses the Basic Encoding Rules (BER).

Parameters:

`out` - BER Output Stream object

`explicit` - Flag indicating explicit tagging should be done

Throws:

`IOException` - Any exception thrown by the underlying OutputStream

`Asn1Exception` - Thrown, if operation is failed.

encode

```
public void encode(Asn1PerOutputStream out)
  throws Asn1Exception,
          java.io.IOException
```

(continued from last page)

This method encodes an unconstrained X.694 order element value using the Packed Encoding Rules (PER). The length and contents components of the message are encoded.

Parameters:

`out` - PER Encode message buffer object

Throws:

`IOException` - Any exception thrown by the Asn1PerOutputStream.
`Asn1Exception` - Thrown, if operation is failed.

com.objsys.asn1j.runtime
Class Asn1XerDecodeBuffer

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XerDecodeBuffer
```

public class **Asn1XerDecodeBuffer**
extends java.lang.Object

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

Field Summary

org.xml.sax.InputSource	mInputSource

Constructor Summary

public	Asn1XerDecodeBuffer(java.lang.String xmlURI) This constructor creates an XER decode buffer.

Method Summary

org.xml.sax.InputSource	getInputSource() This method returns the SAX input source object.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

mInputSource

protected org.xml.sax.InputSource **mInputSource**

Constructors

Asn1XerDecodeBuffer

public **Asn1XerDecodeBuffer**(java.lang.String xmlURI)

This constructor creates an XER decode buffer.

Methods

getInputSource

`public org.xml.sax.InputSource` **`getInputSource`**`()`

This method returns the SAX input source object.

Returns:

SAX input source object

com.objsys.asn1j.runtime
Interface Asn1XerDecodeMethod

public interface **Asn1XerDecodeMethod**

Interface for invoking XER decode method on an object. The method decodes into the host object. Most, but not all, generated classes will implement this interface, which serves as a signal to open type decoding for how to decode a type.

Method Summary

`abstract void`	`decode(java.lang.Object reader,boolean asGroup)` Decode XER.

Methods

decode

```
public abstract void decode(java.lang.Object reader,
         boolean asGroup)
```

Decode XER.

Parameters:

`reader` - The XML pull parser compatible with the generated code.

`asGroup` - Pass TRUE if the current element is the beginning of the content to be decoded. Pass FALSE if the current element's content is the content to be decoded.

com.objsys.asn1j.runtime
Interface Asn1XerDecoder

public interface **Asn1XerDecoder**

Interface for obtaining decoded XER data. This interface is implemented by the Decoder class used for decoding enumerated types, which is generated as a nested class in enumerated type classes.

Method Summary

`abstract Asn1Type`	`decode(java.lang.Object reader,boolean asGroup)` Decode XER.

Methods

decode

```
public abstract Asn1Type decode(java.lang.Object reader,
         boolean asGroup)
```

Decode XER.

Parameters:

`reader` - The XML pull parser compatible with the generated code.

`asGroup` - Pass TRUE if the current element is the beginning of the content to be decoded. Pass FALSE if the current element's content is the content to be decoded.

Returns:

The decoded object.

com.objsys.asn1j.runtime
Class Asn1XerElemInfo

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XerElemInfo
```

public class **Asn1XerElemInfo**
extends java.lang.Object

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

Constructor Summary

public	Asn1XerElemInfo(java.lang.String[] names,boolean optional,int id)

Method Summary

int	getID()
boolean	isOptional()
boolean	matches(java.lang.String name)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XerElemInfo

```
public Asn1XerElemInfo(java.lang.String[] names,
                       boolean optional,
                       int id)
```

Methods

matches

```
public boolean matches(java.lang.String name)
```

(continued from last page)

getID

```
public int getID()
```

isOptional

```
public boolean isOptional()
```

com.objsys.asn1j.runtime
Class Asn1XerEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeByteBuffer
               ┗-com.objsys.asn1j.runtime.Asn1XerEncodeBuffer
```

All Implemented Interfaces:
Asn1XerEncoder

---

public class **Asn1XerEncodeBuffer**
extends Asn1EncodeByteBuffer
implements Asn1XerEncoder

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

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeByteBuffer`
`mByteIndex`, `mData`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`
`INITIAL_SIZE`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

`public`	`Asn1XerEncodeBuffer()` The default constructor creates an XER encode buffer object with the default initial size and canonical set to false.
`public`	`Asn1XerEncodeBuffer(boolean canonical)` The parameterized constructor creates an XER encode buffer object with default initial size and canonical set to the given values.
`public`	`Asn1XerEncodeBuffer(boolean canonical,int size)` The parameterized constructor creates an XER encode buffer object with initial size increment and canonical set to the given values.

Method Summary

void	`binDump(java.io.PrintStream out,java.lang.String varName)` This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.
void	`copy(java.lang.String value)` This method copies a character string to the encode buffer.
void	`decrLevel()` This method decrements the element nesting level counter.
void	`encodeBinStrValue(byte[] bits,int nbits)` This method encodes XML binary string data
void	`encodeByte(byte value)` This method is used to encode a single byte to the encode buffer.
void	`encodeData(java.lang.String value)` This method encodes XML string data
void	`encodeEmptyElement(java.lang.String elemName)` This method encodes an XML empty element tag
void	`encodeEndDocument()` This method encodes standard trailor information at the end of the XML document.
void	`encodeEndElement(java.lang.String elemName)` This method encodes an XML end element tag
void	`encodeHexStrValue(byte[] data)` This method encodes XML hexadecimal string data
void	`encodeNamedValue(java.lang.String valueName,java.lang.String elemName)` This method encodes an XML named value (with start and end tags)
void	`encodeNamedValueElement(java.lang.String elemName)` This method encodes an XML named value element tag
void	`encodeRealValue(double value,java.lang.String elemName)` This method encodes an XML REAL (double) value (with start and end tags).
void	`encodeStartDocument()` This method encodes standard header information at the beginning of the XML document.
void	`encodeStartElement(java.lang.String elemName)` This method encodes an XML start element tag
int	`getState()` This method gets the state of the buffer.
void	`incrLevel()` This method increments the element nesting level counter.

void	indent() This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the encode buffer.
boolean	isCanonical()
void	setCanonical(boolean value) This method sets the canonical encoding flag to the given value.
void	setState(int stat) This method sets the state of the buffer.

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeByteBuffer
checkSize, copy, copy, copy, copyUnsafe, getBuffer, getByteArrayInputStream, getMsgCopy, getMsgLength, initBuffer, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XerEncodeBuffer

public **Asn1XerEncodeBuffer**()

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

Asn1XerEncodeBuffer

public **Asn1XerEncodeBuffer**(boolean canonical)

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

Parameters:

(continued from last page)

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

Asn1XerEncodeBuffer

```
public Asn1XerEncodeBuffer(boolean canonical,
                           int size)
```

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

Parameters:

`canonical` - Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.
`size` - The initial size in bytes of an encode buffer. If this parameter is set to zero, the default size will be used.

Methods

binDump

```
public void binDump(java.io.PrintStream out,
         java.lang.String varName)
```

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

copy

```
public void copy(java.lang.String value)
  throws Asn1Exception
```

This method copies a character string to the encode buffer.

Parameters:

`value` - The string value to copy

decrLevel

```
public void decrLevel()
```

This method decrements the element nesting level counter.

encodeByte

```
public void encodeByte(byte value)
```

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

Parameters:

`value` - The byte value to copy

encodeData

```
public void encodeData(java.lang.String value)
  throws Asn1Exception
```

This method encodes XML string data

Parameters:

`value` - String value to encode

(continued from last page)

encodeBinStrValue

```
public void encodeBinStrValue(byte[] bits,
         int nbits)
  throws Asn1Exception
```

This method encodes XML binary string data

Parameters:

`bits` - Bit string to encode

---

encodeHexStrValue

```
public void encodeHexStrValue(byte[] data)
  throws Asn1Exception
```

This method encodes XML hexadecimal string data

Parameters:

`data` - Data to encode

---

encodeEndDocument

```
public void encodeEndDocument()
  throws Asn1Exception
```

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

---

encodeEndElement

```
public void encodeEndElement(java.lang.String elemName)
  throws Asn1Exception
```

This method encodes an XML end element tag

---

encodeStartDocument

```
public void encodeStartDocument()
  throws Asn1Exception
```

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

---

encodeStartElement

```
public void encodeStartElement(java.lang.String elemName)
  throws Asn1Exception
```

This method encodes an XML start element tag

---

encodeEmptyElement

```
public void encodeEmptyElement(java.lang.String elemName)
  throws Asn1Exception
```

This method encodes an XML empty element tag

---

encodeNamedValueElement

```
public void encodeNamedValueElement(java.lang.String elemName)
  throws Asn1Exception
```

This method encodes an XML named value element tag

Parameters:

`elemName` - The name of element.

(continued from last page)

Throws:
`Asn1Exception` - Thrown, if operation is failed.

encodeRealValue

```
public void encodeRealValue(double value,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

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

Throws:
`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

getState

```
public int getState()
```

This method gets the state of the buffer.

Returns:
Buffer Stat

incrLevel

```
public void incrLevel()
```

This method increments the element nesting level counter.

indent

```
public void indent()
  throws Asn1Exception
```

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

isCanonical

```
public boolean isCanonical()
```

See Also:
`Asn1XmlXerEncoder`

setCanonical

```
public void setCanonical(boolean value)
```

This method sets the canonical encoding flag to the given value.

Parameters:
`value` - Canonical encoding flag.

(continued from last page)

setState

```
public void setState(int stat)
```

This method sets the state of the buffer.

Parameters:

`stat` - Buffer Stat

---

encodeNamedValue

```
public void encodeNamedValue(java.lang.String valueName,
         java.lang.String elemName)
  throws Asn1Exception
```

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

com.objsys.asn1j.runtime
Interface Asn1XerEncoder

All Superinterfaces:
Asn1XmlXerEncoder

All Known Implementing Classes:
Asn1XerOutputStream, Asn1XerEncodeBuffer

---

public interface **Asn1XerEncoder**
extends Asn1XmlXerEncoder

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

Field Summary

static final int	XERDATA Value: **2**
static final int	XEREND Value: **3**
static final int	XERINDENT Value: **3**
static final int	XERINIT Value: **0**
static final int	XERSTART Value: **1**

Method Summary

abstract void	encodeEmptyElement(java.lang.String elemName) This method encodes an XML empty element tag.
abstract void	encodeEndElement(java.lang.String elemName) This method encodes an XML end element tag.
abstract void	encodeNamedValue(java.lang.String valueName,java.lang.String elemName) This method encodes an XML named value (with start and end tags).
abstract void	encodeRealValue(double valueName,java.lang.String elemName) This method encodes an XML REAL (double) value (with start and end tags).
abstract void	encodeStartElement(java.lang.String elemName) This method encodes an XML start element tag.

abstract int	getState() This method gets the state of the buffer.
abstract void	setState(int stat) This method sets the state of the buffer.

Methods inherited from interface com.objsys.asn1j.runtime.Asn1XmlXerEncoder
copy, copy, copy, copy, decrLevel, encodeBinStrValue, encodeData, encodeEndDocument, encodeHexStrValue, encodeNamedValueElement, encodeStartDocument, getContext, incrLevel, indent, isCanonical

Fields

XERINDENT

public static final int **XERINDENT**

XERINIT

public static final int **XERINIT**

XERSTART

public static final int **XERSTART**

XERDATA

public static final int **XERDATA**

XEREND

public static final int **XEREND**

Methods

encodeStartElement

```
public abstract void encodeStartElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML start element tag.

Parameters:

`elemName` - The name of element.

Throws:

(continued from last page)

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeEndElement

```
public abstract void encodeEndElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML end element tag.

Parameters:
`elemName` - The name of element.

Throws:
`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeEmptyElement

```
public abstract void encodeEmptyElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML empty element tag.

Parameters:
`elemName` - The name of element.

Throws:
`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValue

```
public abstract void encodeNamedValue(java.lang.String valueName,
          java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:
`valueName` - The name of value.
`elemName` - The name of element.

Throws:
`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeRealValue

```
public abstract void encodeRealValue(double valueName,
          java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

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

(continued from last page)

Throws:

`IOException -` If I/O error occurs.
`Asn1Exception -` Thrown, if operation is failed.

getState

`public abstract int` **`getState`**`()`

This method gets the state of the buffer.

Returns:

Buffer Stat

setState

`public abstract void` **`setState`**`(int stat)`

This method sets the state of the buffer.

Parameters:

`stat -` Buffer Stat

com.objsys.asn1j.runtime
Class Asn1XerOpenType

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1OctetString
         ┗-com.objsys.asn1j.runtime.Asn1OpenType
            ┗-com.objsys.asn1j.runtime.Asn1XerOpenType
```

Deprecated. *Use Asn1OpenType instead. You can use the constructors where you specify the encoding, and specify XER if you are populating with byte data that is a UTF-8 XML encoding of the actual value.*

public class **Asn1XerOpenType**
extends Asn1OpenType

This is a container class for holding the an ASN.1 open type value. This is a special version of the class that is only generated for the XER/XML encoding rules. The data held in objects of this type should be UTF-8 encoded XML

Fields inherited from class com.objsys.asn1j.runtime.Asn1OpenType
BER, dataEncoding, EDATAMSG, JSON, mEncodeBuffer, mLength, OER, PER, UNKNOWN, XER

Fields inherited from class com.objsys.asn1j.runtime.Asn1OctetString
TAG, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1XerOpenType() **Deprecated.**
public	Asn1XerOpenType(byte[] data) **Deprecated.**
public	Asn1XerOpenType(byte[] data,int offset,int nbytes) **Deprecated.**
public	Asn1XerOpenType(Asn1EncodeBuffer buffer) **Deprecated.**

Methods inherited from class com.objsys.asn1j.runtime.Asn1OpenType
decode, decode, decode, decode, decode72, decodeExtension, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode72, encodeAsExtension, encodeAsExtension, encodeAsExtension, getAsn1TypeName, getCharData, getDataEncoding, getSaxHandler, getSaxHandler, setBinaryData, setCharData, setCharData, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1OctetString
compareTo, decode, decode, decode, decode, decode, decode, decode, decode, decodeAsBase64, decodeAsHex, decodeContent, decodeRemainingBits, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsBase64, encodeAsHex, encodeAttribute, encodeBase64Binary, encodeContent, equals, equals, equals, getAsn1TypeName, getLength, getMderLength, hashCode, toInputStream, toString

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XerOpenType

public **Asn1XerOpenType**()

Deprecated.

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

Asn1XerOpenType

public **Asn1XerOpenType**(byte[] data)

Deprecated.

This constructor initializes an open type from the given byte array. The array is assumed to contain a previously encoded message component. The data is assumed to be UTF-8 encoded XML. It should represent an XML encoding of the actual type.

Parameters:

data - Byte array containing a previously encoded value.

(continued from last page)

Asn1XerOpenType

```
public Asn1XerOpenType(byte[] data,
                       int offset,
                       int nbytes)
```

Deprecated.

This constructor initializes the open type from a portion of the given byte array. A new byte array is created starting at the given offset and consisting of the given number of bytes. The data is assumed to be UTF-8 encoded XML. It should represent an XML encoding of the actual type.

Parameters:

`data` - Byte array containing a previously encoded value.
`offset` - Starting offset in data from which to copy bytes
`nbytes` - Number of bytes to copy from target array

Asn1XerOpenType

```
public Asn1XerOpenType(Asn1EncodeBuffer buffer)
```

Deprecated.

Parameters:

`buffer`

See Also:

`Asn1OpenType.Asn1OpenType(Asn1EncodeBuffer)()`

com.objsys.asn1j.runtime
Class Asn1XerOutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-com.objsys.asn1j.runtime.Asn1OutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1XerOutputStream
```

All Implemented Interfaces:
Asn1XerEncoder, java.io.Flushable, java.io.Closeable

---

public class **Asn1XerOutputStream**
extends Asn1OutputStream
implements java.io.Closeable, java.io.Flushable, Asn1XerEncoder

This class implements the output stream to encode ASN.1 messages as specified in the XML Encoding Rules (XER) as specified in the ITU-T X.693 standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OutputStream`
`os`

Constructor Summary

public	`Asn1XerOutputStream(java.io.OutputStream os)` This constructor creates a buffered XER output stream object with default size of buffer.
public	`Asn1XerOutputStream(java.io.OutputStream os,boolean canonical,int bufSize)` This constructor creates a buffered XER output stream object.

Method Summary

void	`copy(byte value)` This method is used to copy a single byte to the output stream.
void	`copy(byte[] value)` This method copies multiple bytes to the output stream.
void	`copy(byte[] value,int off,int len)` This method copies multiple bytes to the output stream.
void	`copy(java.lang.String value)` This method copies a character string to the output stream.
void	`decrLevel()` This method decrements the element nesting level counter.

void	`encodeBinStrValue(byte[] bits,int nbits)` This method encodes XML binary string data
void	`encodeByte(byte value)` This method is used to encode a single byte to the output stream.
void	`encodeData(java.lang.String value)` This method encodes XML string data
void	`encodeEmptyElement(java.lang.String elemName)` This method encodes an XML empty element tag.
void	`encodeEndDocument()` This method encodes standard trailor information at the end of the XML document.
void	`encodeEndElement(java.lang.String elemName)` This method encodes an XML end element tag.
void	`encodeHexStrValue(byte[] data)` This method encodes XML hexadecimal string data
void	`encodeNamedValue(java.lang.String valueName,java.lang.String elemName)` This method encodes an XML named value (with start and end tags).
void	`encodeNamedValueElement(java.lang.String elemName)` This method encodes an XML named value element tag.
void	`encodeRealValue(double value,java.lang.String elemName)` This method encodes an XML REAL (double) value (with start and end tags).
void	`encodeStartDocument()` This method encodes standard header information at the beginning of the XML document.
void	`encodeStartElement(java.lang.String elemName)` This method encodes an XML start element tag.
int	`getState()` This method gets the state of the buffer.
void	`incrLevel()` This method increments the element nesting level counter.
void	`indent()` This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the output stream.
boolean	`isCanonical()`
void	`setCanonical(boolean value)` This method sets the canonical encoding flag to the given value.
void	`setState(int stat)` This method sets the state of the buffer.

void	write(java.lang.String value) This method copies a character string to the output stream.

Methods inherited from class com.objsys.asn1j.runtime.Asn1OutputStream
close, flush, getContext, write, write, write, write2Bytes, write4Bytes

Methods inherited from class java.io.OutputStream
close, flush, write, write, write

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XerOutputStream

```
public Asn1XerOutputStream(java.io.OutputStream os)
```

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

Parameters:

`os` - The underlying OutputStream object.

Asn1XerOutputStream

```
public Asn1XerOutputStream(java.io.OutputStream os,
                           boolean canonical,
                           int bufSize)
```

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

Parameters:

`os` - The underlying OutputStream object.
`canonical` - Boolean indicating a canonical or non-canonical encoding should be produced as defined in the X.693 standard.
`bufSize` - The buffer size. If it is 0 then the output stream is used as unbuffered.

Methods

copy

```
public void copy(byte value)
  throws java.io.IOException
```

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

Parameters:

`value` - The byte value to copy

(continued from last page)

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

---

copy

```
public void copy(byte[] value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Array of bytes to copy to the output stream

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

copy

```
public void copy(byte[] value,
          int off,
          int len)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

copy

```
public void copy(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method copies a character string to the output stream.

Parameters:

`value` - The string value to copy

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

write

```
public void write(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method copies a character string to the output stream.

Parameters:

(continued from last page)

`value` - The string value to copy

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

decrLevel

```
public void decrLevel()
```

This method decrements the element nesting level counter.

encodeByte

```
public void encodeByte(byte value)
  throws java.io.IOException
```

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

Parameters:

`value` - The byte value to copy

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeData

```
public void encodeData(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML string data

Parameters:

`value` - String value to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encodeBinStrValue

```
public void encodeBinStrValue(byte[] bits,
          int nbits)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML binary string data

Parameters:

`bits` - Bit string to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

encodeHexStrValue

```
public void encodeHexStrValue(byte[] data)
  throws java.io.IOException,
         Asn1Exception
```

(continued from last page)

This method encodes XML hexadecimal string data

Parameters:

`data` - Data to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeEndDocument

```
public void encodeEndDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeEndElement

```
public void encodeEndElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML end element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeStartDocument

```
public void encodeStartDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeStartElement

```
public void encodeStartElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML start element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeEmptyElement

```
public void encodeEmptyElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML empty element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValueElement

```
public void encodeNamedValueElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML named value element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValue

```
public void encodeNamedValue(java.lang.String valueName,
          java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeRealValue

```
public void encodeRealValue(double value,
          java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - If I/O error occurs.

(continued from last page)

`Asn1Exception` - Thrown, if operation is failed.

getState

```
public int getState()
```

This method gets the state of the buffer.

Returns:

Buffer Stat

incrLevel

```
public void incrLevel()
```

This method increments the element nesting level counter.

indent

```
public void indent()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

isCanonical

```
public boolean isCanonical()
```

See Also:

`Asn1XmlXerEncoder`

setCanonical

```
public void setCanonical(boolean value)
```

This method sets the canonical encoding flag to the given value.

Parameters:

`value` - Canonical encoding flag.

setState

```
public void setState(int stat)
```

This method sets the state of the buffer.

Parameters:

`stat` - Buffer Stat

com.objsys.asn1j.runtime
Class Asn1XerSaxHandler

```
java.lang.Object
   ┗-org.xml.sax.helpers.DefaultHandler
      ┗-com.objsys.asn1j.runtime.Asn1XerSaxHandler
```

Direct Known Subclasses:
Asn1XmlSaxSimpleType, XsdAnySaxHandler, SaxHandler

---

public class **Asn1XerSaxHandler**
extends org.xml.sax.helpers.DefaultHandler

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

Field Summary

boolean	mConsumedStartElement
int	mCurrElemID
int	mCurrState
int	mLevel
int	mStartLevel
java.lang.String	mXMLElemName
final int	XERDATA Value: **2**
final int	XEREND Value: **3**
final int	XERINIT Value: **0**
final int	XERSTART Value: **1**
final int	XERUNKNOWN Value: **-1**

Constructor Summary

protected	Asn1XerSaxHandler()

Method Summary

boolean	consumeStartElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName,org.xml.sax.Attributes atts) This method should be used in preference to invoking startElement directly when a parent SAX handler is delegating to a child SAX handler.
void	endGroup() This event method should be invoked when the group being decoded by this handler is decided to be complete.
void	error(org.xml.sax.SAXParseException exception)
void	fatalError(org.xml.sax.SAXParseException exception)
int	getState()
void	init(int startLevel)
boolean	isComplete() Returns true when the SAX handler has finished decoding the group.
boolean	isDecodingAsGroup() Return true if this SAX handler is decoding a group of elements rather than a single element (and possibly its children elements).
boolean	matchXMLElemName(java.lang.String elemName)
void	setComplete() Invoke this method to mark this SAX handler as being complete.
void	setLevel(int startLevel)
void	setXMLElemName(java.lang.String value)
void	warning(org.xml.sax.SAXParseException exception)

Methods inherited from class `org.xml.sax.helpers.DefaultHandler`

`characters, endDocument, endElement, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator, skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

XERUNKNOWN

protected final int **XERUNKNOWN**

XERINIT

protected final int **XERINIT**

XERSTART

protected final int **XERSTART**

XERDATA

protected final int **XERDATA**

XEREND

protected final int **XEREND**

mCurrState

protected int **mCurrState**

mCurrElemID

protected int **mCurrElemID**

mLevel

protected int **mLevel**

mStartLevel

protected int **mStartLevel**

mXMLElemName

protected java.lang.String **mXMLElemName**

mConsumedStartElement

```
protected boolean mConsumedStartElement
```

Constructors

Asn1XerSaxHandler

```
protected Asn1XerSaxHandler()
```

Methods

getState

```
public int getState()
```

init

```
public void init(int startLevel)
```

endGroup

```
public void endGroup()
  throws org.xml.sax.SAXException
```

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

consumeStartElement

```
public final boolean consumeStartElement(java.lang.String namespaceURI,
         java.lang.String localName,
         java.lang.String qName,
         org.xml.sax.Attributes atts)
  throws org.xml.sax.SAXException
```

This method should be used in preference to invoking startElement directly when a parent SAX handler is delegating to a child SAX handler. Using this method gives the startElement method the opportunity to set the mConsumedStartEvent flag to false to signal that the given element does not belong to the group and that the group is complete. This method invokes the startElement event and returns the resulting value of the mConsumedStartElement flag (true, unless the startElement method explicitly sets it to false). If mConsumedStartElement is set to false, this method will invoke setComplete, marking the handler complete and triggering the endGroup event, if it has not already fired. If the startElement method is certain that some other element is required instead of the one given, it is preferable to throw Asn1XmlSaxUnexpElemExc to indicate this. Otherwise, if the given element does not belong to the group being decoded, mConsumedStartElement can be be set false and the element ignored. It is then up to the parent SAX handler to recognize the element as part of a different group or report it as an unexpected element. The startElement method should not set mConsumedStartEvent false except when mLevel <= mStartLevel, since this is a precondition for setComplete.

Returns:

true if the startElement method consumed the given element or false if not.

Throws:

`Asn1XmlUnexpElemExc` - if certain that some other element is expected.

isComplete

```
public final boolean isComplete()
```

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

isDecodingAsGroup

```
public final boolean isDecodingAsGroup()
```

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

setComplete

```
public final void setComplete()
  throws org.xml.sax.SAXException
```

Invoke this method to mark this SAX handler as being complete. This method will trigger the endGroup event, if it has not already fired. This is the preferred way for a subclass to trigger the endGroup event, rather than to invoke endGroup directly (mainly for consistency). Subclasses should consider invoking this method in the endElement event when either of the following conditions obtain: - mLevel returns to 0 (indicating the group must be complete) - The group is self-delimiting and known to be complete on that basis This method should only be invoked when mLevel <= mStartLevel. Otherwise, the handler could not possibly be complete (note that mLevel == mStartLevel does not entail the handler is complete, because this is a normal state of affairs between elements when decoding a group).

Throws:

`IllegalStateException` - if mLevel != mStartLevel.

setLevel

```
public void setLevel(int startLevel)
```

warning

```
public void warning(org.xml.sax.SAXParseException exception)
  throws org.xml.sax.SAXException
```

error

```
public void error(org.xml.sax.SAXParseException exception)
  throws org.xml.sax.SAXException
```

fatalError

```
public void fatalError(org.xml.sax.SAXParseException exception)
  throws org.xml.sax.SAXException
```

matchXMLElemName

```
public boolean matchXMLElemName(java.lang.String elemName)
```

(continued from last page)

setXMLElemName

```
public void setXMLElemName(java.lang.String value)
```

com.objsys.asn1j.runtime
Class Asn1XerUtil

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XerUtil
```

public class **Asn1XerUtil**
extends java.lang.Object

Constructor Summary

public	Asn1XerUtil()

Method Summary

static void	encodeReal(Asn1XerEncoder buffer,double value,java.lang.String elemName) This method encodes an ASN.1 real value using the XML encoding rules (XER).

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XerUtil

```
public Asn1XerUtil()
```

Methods

encodeReal

```
public static void encodeReal(Asn1XerEncoder buffer,
         double value,
         java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

com.objsys.asn1j.runtime
Class Asn1XmlAnyElem

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1Type
      ┗-com.objsys.asn1j.runtime.Asn1CharString
         ┗-com.objsys.asn1j.runtime.Asn1UTF8String
            ┗-com.objsys.asn1j.runtime.Asn1XmlAnyElem
```

public class **Asn1XmlAnyElem**
extends Asn1UTF8String

This is a container class for holding the components of an XSD any element wildcard (xsd:any) type. The string contained within this object must be a well-formed XML fragment.

Fields inherited from class com.objsys.asn1j.runtime.Asn1UTF8String
TAG

Fields inherited from class com.objsys.asn1j.runtime.Asn1CharString
mStringBuffer, value

Fields inherited from class com.objsys.asn1j.runtime.Asn1Type
BIT_STRING, BMPString, BOOLEAN, DATE, DATE_TIME, DURATION, ENUMERATED, EOC, EXTERNAL, GeneralString, GeneralTime, GraphicString, IA5String, INTEGER, mNonParameterizedTypeName, NULL, NumericString, OBJECT_IDENTIFIER, ObjectDescriptor, OCTET_STRING, OID_IRI, OpenType, PrintableString, REAL, RELATIVE_OID_IRI, RelativeOID, SEQUENCE, SET, T61String, TeletexString, TIME, TIME_OF_DAY, UniversalString, UTCTime, UTF8String, VideotexString, VisibleString

Constructor Summary

public	Asn1XmlAnyElem() The default constructor initializes the underlying UTF-8 string object.
public	Asn1XmlAnyElem(java.lang.String value) This constructor sets the underlying XML string value.

Method Summary

void	encode(Asn1XmlEncoder buffer,java.lang.String elemName,java.lang.String nsPrefix) This method encodes the string in XML format.
Asn1XerSaxHandler	getSaxHandler()

Methods inherited from class com.objsys.asn1j.runtime.Asn1UTF8String

decode, decode, decode, decode, decode, decode, decodeUTF8, decodeUTF8, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, setAnyAttribute

Methods inherited from class com.objsys.asn1j.runtime.Asn1CharString
decode, decode, decode, decode, decodeByteToChar, decodeXER, decodeXML, encode, encode, encode, encode, encode, encode, encode, equals, equals, getAsn1TypeName, getLength, hashCode, toString, validate

Methods inherited from class com.objsys.asn1j.runtime.Asn1Type
_setKey, clone, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decode, decodeXML, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encode, encodeAsOpenType, equals, getAsn1TypeName, getLength, getNonParameterizedTypeName, getTypeName, hashCode, indent, indent, isOpenType, matchTag, matchTag, pdiag, print, print, print, setNonParameterizedTypeName, setOpenType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XmlAnyElem

```
public Asn1XmlAnyElem()
```

The default constructor initializes the underlying UTF-8 string object.

Asn1XmlAnyElem

```
public Asn1XmlAnyElem(java.lang.String value)
```

This constructor sets the underlying XML string value. It must contain well-formed XML text unless encoding is to be done using an explicit element name.

Parameters:

`value` - String containing well-formed XML text.

Methods

encode

```
public void encode(Asn1XmlEncoder buffer,
                   java.lang.String elemName,
                   java.lang.String nsPrefix)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes the string in XML format. It does a basic well-formedness check on the content to make sure the string contains a valid XML fragment. It then copies the text to the output buffer or stream.

Parameters:

`buffer` - XML encode buffer or stream object.

`elemName` - Local name of the element (if any).

(continued from last page)

`nsPrefix` - Namespace prefix (if any).

getSaxHandler

`public Asn1XerSaxHandler` **`getSaxHandler`**`()`

com.objsys.asn1j.runtime
Class Asn1XmlAnyElem.XsdAnySaxHandler

```
java.lang.Object
   ┗-org.xml.sax.helpers.DefaultHandler
      ┗-com.objsys.asn1j.runtime.Asn1XerSaxHandler
         ┗-com.objsys.asn1j.runtime.Asn1XmlAnyElem.XsdAnySaxHandler
```

public class **Asn1XmlAnyElem.XsdAnySaxHandler**
extends Asn1XerSaxHandler

Fields inherited from class com.objsys.asn1j.runtime.Asn1XerSaxHandler
mConsumedStartElement, mCurrElemID, mCurrState, mLevel, mStartLevel, mXMLElemName, XERDATA, XEREND, XERINIT, XERSTART, XERUNKNOWN

Method Summary

void	characters(char[] ch,int start,int length)
void	endElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName)
void	startElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName,org.xml.sax.Attributes atts)

Methods inherited from class com.objsys.asn1j.runtime.Asn1XerSaxHandler
consumeStartElement, endGroup, error, fatalError, getState, init, isComplete, isDecodingAsGroup, matchXMLElemName, setComplete, setLevel, setXMLElemName, warning

Methods inherited from class org.xml.sax.helpers.DefaultHandler
characters, endDocument, endElement, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator, skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

(continued from last page)

startElement

```
public void startElement(java.lang.String namespaceURI,
          java.lang.String localName,
          java.lang.String qName,
          org.xml.sax.Attributes atts)
  throws org.xml.sax.SAXException
```

---

characters

```
public void characters(char[] ch,
          int start,
          int length)
  throws org.xml.sax.SAXException
```

---

endElement

```
public void endElement(java.lang.String namespaceURI,
          java.lang.String localName,
          java.lang.String qName)
  throws org.xml.sax.SAXException
```

com.objsys.asn1j.runtime
Class Asn1XmlEncodeBuffer

```
java.lang.Object
   ┗-com.objsys.asn1j.runtime.Asn1MessageBufferBase
      ┗-com.objsys.asn1j.runtime.Asn1MessageBuffer
         ┗-com.objsys.asn1j.runtime.Asn1EncodeBuffer
            ┗-com.objsys.asn1j.runtime.Asn1EncodeByteBuffer
               ┗-com.objsys.asn1j.runtime.Asn1XmlEncodeBuffer
```

All Implemented Interfaces:
Asn1XmlEncoder

---

public class **Asn1XmlEncodeBuffer**
extends Asn1EncodeByteBuffer
implements Asn1XmlEncoder

This class handles the encoding of ASN.1 messages as specified in the XML Encoding (non-XER) as specified in the XML schema standard.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeByteBuffer`
`mByteIndex`, `mData`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1EncodeBuffer`
`INITIAL_SIZE`

Fields inherited from class `com.objsys.asn1j.runtime.Asn1MessageBufferBase`
`context`, `mTypeCode`

Constructor Summary

public	`Asn1XmlEncodeBuffer()` The default constructor creates an XML encode buffer object with the default initial size and canonical set to false.
public	`Asn1XmlEncodeBuffer(int size)` The parameterized constructor creates an XML encode buffer object with initial size set to the given value.

Method Summary

void	`binDump(java.io.PrintStream out,java.lang.String varName)` This method dumps the encoded message in a human-readable format showing a bit trace of all fields to the given print output stream.

void	`copy(java.lang.String value)` This method copies a character string to the encode buffer.
void	`decrLevel()` This method decrements the element nesting level counter.
void	`encodeAttr(java.lang.String qname,java.lang.String value)` This method encodes an XML attribute value.
void	`encodeBinStrValue(byte[] bits,int nbits)` This method encodes XML binary string data
void	`encodeData(java.lang.String value)` This method encodes XML string data
void	`encodeDoubleValue(double value,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML REAL (double) value (with start and end tags).
void	`encodeEmptyElement(java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML empty element tag
void	`encodeEndDocument()` This method encodes standard trailor information at the end of the XML document.
void	`encodeEndElement(java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML end element tag
void	`encodeHexStrValue(byte[] data)` This method encodes XML hexadecimal string data
void	`encodeNamedValue(java.lang.String valueName,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML named value (with start and end tags)
void	`encodeNamedValueElement(java.lang.String elemName)` This method encodes an XML named value element.
void	`encodeStartDocument()` This method encodes standard header information at the beginning of the XML document.
void	`encodeStartElement(java.lang.String elemName,java.lang.String nsPrefix,boolean terminate)` This method encodes an XML start element tag.
void	`encodeXSIAttrs()` This method encodes XSI attributes.
Asn1XmlEncodeHelper	`getHelper()` This method returns a reference to the internal helper object.
void	`incrLevel()` This method increments the element nesting level counter.

void	indent() This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the encode buffer.
boolean	isCanonical()
void	setCanonical(boolean value) This method sets the canonical encoding flag to the given value.
void	setIndent(int value) This method sets the number of spaces per indentation.
void	setMixedContent(boolean value) This method sets the mixed content flag which indicates that this buffer will be used to encode mixed content.
void	setTermStart(boolean value) This method sets the start element termination required flag.
void	setXSIAttrs(Asn1XmlXSIAttrs value) This method sets the XSI attributes object to the given value.

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeByteBuffer
checkSize, copy, copy, copy, copyUnsafe, getBuffer, getByteArrayInputStream, getMsgCopy, getMsgLength, initBuffer, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1EncodeBuffer
binDump, binDump, copy, copy, copy, encodeIntSigned, encodeIntUnsigned, getByteArrayInputStream, getInputStream, getMinimalOctetsSigned, getMinimalOctetsUnsigned, getMsgCopy, getMsgLength, getOutputStream, hexDump, hexDump, reset, write

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBuffer
addNamedEventHandler, getEventHandlerListCount, getInputStream, hasEventHandlers, invokeCharacters, invokeEndElement, invokeStartElement, setEventHandlerList

Methods inherited from class com.objsys.asn1j.runtime.Asn1MessageBufferBase
getContext, hexDump, hexDump, setKey, setTypeCode

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

(continued from last page)

Asn1XmlEncodeBuffer

```
public Asn1XmlEncodeBuffer()
```

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

Asn1XmlEncodeBuffer

```
public Asn1XmlEncodeBuffer(int size)
```

The parameterized constructor creates an XML encode buffer object with initial size set to the given value.

Parameters:

`size` - The initial size in bytes of an encode buffer. If this parameter is set to zero, the default increment will be used.

Methods

binDump

```
public void binDump(java.io.PrintStream out,
         java.lang.String varName)
```

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

copy

```
public void copy(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method copies a character string to the encode buffer.

Parameters:

`value` - The string value to copy

decrLevel

```
public void decrLevel()
```

This method decrements the element nesting level counter.

encodeData

```
public void encodeData(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML string data

Parameters:

`value` - String value to encode

encodeAttr

```
public void encodeAttr(java.lang.String qname,
         java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML attribute value.

(continued from last page)

Parameters:

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

encodeBinStrValue

```
public void encodeBinStrValue(byte[] bits,
         int nbits)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML binary string data

Parameters:

`bits` - Bit string to encode

encodeDoubleValue

```
public void encodeDoubleValue(double value,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeEmptyElement

```
public void encodeEmptyElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML empty element tag

Parameters:

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

encodeEndDocument

```
public void encodeEndDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

encodeEndElement

```
public void encodeEndElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML end element tag

(continued from last page)

Parameters:

`elemName` - The name of element.

`nsPrefix` - Namespace prefix.

encodeHexStrValue

```
public void encodeHexStrValue(byte[] data)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML hexadecimal string data

Parameters:

`data` - Data to encode

encodeNamedValue

```
public void encodeNamedValue(java.lang.String valueName,
          java.lang.String elemName,
          java.lang.String nsPrefix)
  throws Asn1Exception,
         java.io.IOException
```

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

encodeNamedValueElement

```
public void encodeNamedValueElement(java.lang.String elemName)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an XML named value element.

Parameters:

`elemName` - The name of element.

encodeStartDocument

```
public void encodeStartDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`Asn1Exception` - Thrown, if operation is failed.

encodeStartElement

```
public void encodeStartElement(java.lang.String elemName,
          java.lang.String nsPrefix,
          boolean terminate)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an XML start element tag.

Parameters:

`elemName` - The name of element.

`nsPrefix` - The namespace prefix of element.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

encodeXSIAttrs

```
public void encodeXSIAttrs()
  throws Asn1Exception,
         java.io.IOException
```

This method encodes XSI attributes.

getHelper

```
public Asn1XmlEncodeHelper getHelper()
```

This method returns a reference to the internal helper object.

Returns:

Reference to mHelper object.

incrLevel

```
public void incrLevel()
```

This method increments the element nesting level counter.

indent

```
public void indent()
  throws Asn1Exception,
         java.io.IOException
```

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

isCanonical

```
public boolean isCanonical()
```

See Also:

`Asn1XmlXerEncoder`

setCanonical

```
public void setCanonical(boolean value)
```

This method sets the canonical encoding flag to the given value.

Parameters:

`value` - Canonical encoding flag.

setIndent

```
public void setIndent(int value)
```

This method sets the number of spaces per indentation. If the value is set to zero, no indentation of the XML data is done.

Parameters:

`value` - Number of spaces per indentation level.

setMixedContent

```
public void setMixedContent(boolean value)
```

(continued from last page)

This method sets the mixed content flag which indicates that this buffer will be used to encode mixed content.

Parameters:

`value` - Boolean value

setTermStart

`public void setTermStart(boolean value)`

This method sets the start element termination required flag.

Parameters:

`value` - Boolean value

setXSIAttrs

`public void setXSIAttrs(Asn1XmlXSIAttrs value)`

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

Parameters:

`value` - XSI attributes object

com.objsys.asn1j.runtime
Class Asn1XmlEncodeHelper

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XmlEncodeHelper
```

public class **Asn1XmlEncodeHelper**
extends java.lang.Object

Constructor Summary

public	`Asn1XmlEncodeHelper(Asn1XmlEncoder encoder)`

Method Summary

void	`encodeByte(byte value)` This method is used to encode a single byte to the output stream.
void	`encodeEmptyElement(java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML empty element tag
void	`encodeEndElement(java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML end element tag
void	`encodeNamespace(Asn1XmlNamespace namespace)` Encode the given namespace on the next start element tag.
void	`encodeNSAttrs()` This method encodes XML namespace attributes in the form 'xmlns[:prefix]="uri"'.
void	`encodeStartElement(java.lang.String elemName,java.lang.String nsPrefix,boolean terminate)` This method encodes an XML start element tag.
Asn1XmlNamespaceCtxt	`getNamespaceCtxt()` This method gets the current namespace context.
void	`setMixedContent(boolean value)` This method sets the mixed content flag which indicates that this buffer will be used to encode mixed content.
void	`setNamespaceCtxt(Asn1XmlNamespaceCtxt nsCtxt,boolean force)` This method sets the namespace context to the given value.
void	`setTermStart(boolean value)` This method sets the start element termination required flag.

void	setXMLState(int value) This method sets the XML state.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XmlEncodeHelper

```
public Asn1XmlEncodeHelper(Asn1XmlEncoder encoder)
```

Methods

encodeByte

```
public void encodeByte(byte value)
  throws java.io.IOException
```

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

Parameters:

`value` - The byte value to copy

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

encodeStartElement

```
public void encodeStartElement(java.lang.String elemName,
         java.lang.String nsPrefix,
         boolean terminate)
  throws Asn1Exception,
         java.io.IOException
```

This method encodes an XML start element tag.

Parameters:

`elemName` - The name of element.

`nsPrefix` - The namespace prefix of element.

Throws:

`Asn1Exception` - Thrown, if operation is failed.

encodeNamespace

```
public void encodeNamespace(Asn1XmlNamespace namespace)
```

Encode the given namespace on the next start element tag.

(continued from last page)

encodeEndElement

```
public void encodeEndElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML end element tag

Parameters:

`elemName` - The name of element.
`nsPrefix` - Namespace prefix.

---

encodeEmptyElement

```
public void encodeEmptyElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML empty element tag

Parameters:

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

---

encodeNSAttrs

```
public void encodeNSAttrs()
  throws Asn1Exception,
         java.io.IOException
```

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

---

getNamespaceCtxt

```
public Asn1XmlNamespaceCtxt getNamespaceCtxt()
```

This method gets the current namespace context.

Returns:

Current namespace context

---

setMixedContent

```
public void setMixedContent(boolean value)
```

This method sets the mixed content flag which indicates that this buffer will be used to encode mixed content.

Parameters:

`value` - Boolean value

---

setNamespaceCtxt

```
public void setNamespaceCtxt(Asn1XmlNamespaceCtxt nsCtxt,
         boolean force)
```

This method sets the namespace context to the given value. It is only set if it was not set previously (i.e. is non-null). To force the variable to be set regardless of current state, set the force flag to true.

Parameters:

`nsCtxt` - Reference to namespace context object.
`force` - Flag to indicate value should be set regardless of current state. If false, table will only be set if reference is currently null.

setTermStart

`public void setTermStart(boolean value)`

This method sets the start element termination required flag.

Parameters:

`value` - Boolean value

setXMLState

`public void setXMLState(int value)`

This method sets the XML state.

Parameters:

`value` - XML state value from Asn1XmlEncoder.

com.objsys.asn1j.runtime
Interface Asn1XmlEncoder

All Superinterfaces:
Asn1XmlXerEncoder

All Known Implementing Classes:
Asn1XmlOutputStream, Asn1XmlEncodeBuffer

---

public interface **Asn1XmlEncoder**
extends Asn1XmlXerEncoder

This is a base interface for encoding of ASN.1 messages as XML as specified in the W3C standard. It is implemented by both the Asn1XmlEncodeBuffer and Asn1XmlOutputStream.

Field Summary

static final int	`XMLDATA` Value: **2**
static final int	`XMLEND` Value: **3**
static final int	`XMLINDENT` Value: **3**
static final int	`XMLINIT` Value: **0**
static final int	`XMLSTART` Value: **1**

Method Summary

abstract void	`encodeAttr(java.lang.String qname,java.lang.String value)` This method encodes an XML attribute value.
abstract void	`encodeDoubleValue(double valueName,java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML REAL (double) value (with start and end tags).
abstract void	`encodeEmptyElement(java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML empty element tag.
abstract void	`encodeEndElement(java.lang.String elemName,java.lang.String nsPrefix)` This method encodes an XML end element tag.

abstract void	encodeNamedValue(java.lang.String valueName,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an XML named value (with start and end tags).
abstract void	encodeStartElement(java.lang.String elemName,java.lang.String nsPrefix,boolean terminate) This method encodes an XML start element and attribute tag.
abstract void	encodeXSIAttrs() This method encodes XSI attributes.
abstract Asn1XmlEncodeHelper	getHelper() This method returns a reference to the internal helper object.
abstract void	setIndent(int value) This method sets the number of spaces per indentation.
abstract void	setXSIAttrs(Asn1XmlXSIAttrs value) This method sets the XSI attributes object to the given value.

Methods inherited from interface com.objsys.asn1j.runtime.Asn1XmlXerEncoder
copy, copy, copy, copy, decrLevel, encodeBinStrValue, encodeData, encodeEndDocument, encodeHexStrValue, encodeNamedValueElement, encodeStartDocument, getContext, incrLevel, indent, isCanonical

Fields

XMLINDENT

public static final int **XMLINDENT**

XMLINIT

public static final int **XMLINIT**

XMLSTART

public static final int **XMLSTART**

XMLDATA

public static final int **XMLDATA**

XMLEND

public static final int **XMLEND**

Methods

encodeAttr

```
public abstract void encodeAttr(java.lang.String qname,
         java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML attribute value.

Parameters:

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

---

encodeStartElement

```
public abstract void encodeStartElement(java.lang.String elemName,
         java.lang.String nsPrefix,
         boolean terminate)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`elemName` - The name of element.
`nsPrefix` - Element namespace prefix value

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeEndElement

```
public abstract void encodeEndElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML end element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeEmptyElement

```
public abstract void encodeEmptyElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`elemName` - The name of element.
`nsPrefix` - Element namespace prefix value

(continued from last page)

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValue

```
public abstract void encodeNamedValue(java.lang.String valueName,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`valueName` - The name of value.
`elemName` - The name of element.
`nsPrefix` - Element namespace prefix value

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeDoubleValue

```
public abstract void encodeDoubleValue(double valueName,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`valueName` - The name of value.
`elemName` - The name of element. If null, then start and end
`nsPrefix` - Element namespace prefix value

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeXSIAttrs

```
public abstract void encodeXSIAttrs()
  throws Asn1Exception,
         java.io.IOException
```

This method encodes XSI attributes.

getHelper

```
public abstract Asn1XmlEncodeHelper getHelper()
```

This method returns a reference to the internal helper object.

Returns:

Reference to mHelper object.

(continued from last page)

setIndent

`public abstract void` **`setIndent`**`(int value)`

This method sets the number of spaces per indentation. If the value is set to zero, no indentation of the XML data is done.

Parameters:

`value` - Number of spaces per indentation level.

---

setXSIAttrs

`public abstract void` **`setXSIAttrs`**`(Asn1XmlXSIAttrs value)`

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

Parameters:

`value` - XSI attributes object

com.objsys.asn1j.runtime
Class Asn1XmlMisReqElemExc

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1XmlMisReqElemExc
```

public class **Asn1XmlMisReqElemExc**
extends Asn1Exception

This class defines the 'missing required element' exception. This exception is thrown after a sequence of all typed subtree is parsed and it is determined that elements declared to be required in the schema are not present.

Constructor Summary

public	`Asn1XmlMisReqElemExc(java.lang.String javaElemName)` This constructor creates an exception object with a textual message describing the missing element value.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1XmlMisReqElemExc

`public` **`Asn1XmlMisReqElemExc`**`(java.lang.String javaElemName)`

This constructor creates an exception object with a textual message describing the missing element value.

Parameters:

`javaElemName` - Name of Java element that is not present.

com.objsys.asn1j.runtime
Class Asn1XmlNamespace

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XmlNamespace
```

public class **Asn1XmlNamespace**
extends java.lang.Object

This class maps an XML namespace prefix to a URI.

Constructor Summary

public	`Asn1XmlNamespace(java.lang.String prefix,java.lang.String uri)` The constructor creates a mapping between a namespace URI and a prefix.

Method Summary

`java.lang.String`	`getPrefix()` This method is used to get the prefix value.
`java.lang.String`	`getURI()` This method is used to get the URI value.
`boolean`	`isPrefixEqual(java.lang.String prefix)` This method is used to test the given prefix for a match.
`boolean`	`isURIEqual(java.lang.String uri)` This method is used to test the given URI for a match.
`void`	`setPrefix(java.lang.String prefix)` This method is used to set the prefix value.
`void`	`setURI(java.lang.String uri)` This method is used to get the URI value.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1XmlNamespace

```
public Asn1XmlNamespace(java.lang.String prefix,
                        java.lang.String uri)
```

(continued from last page)

The constructor creates a mapping between a namespace URI and a prefix.

Parameters:

`prefix` - XML namespace prefix
`uri` - XML namespace URI

Methods

getPrefix

`public java.lang.String` **`getPrefix`**`()`

This method is used to get the prefix value.

getURI

`public java.lang.String` **`getURI`**`()`

This method is used to get the URI value.

isPrefixEqual

`public boolean` **`isPrefixEqual`**`(java.lang.String prefix)`

This method is used to test the given prefix for a match.

isURIEqual

`public boolean` **`isURIEqual`**`(java.lang.String uri)`

This method is used to test the given URI for a match.

setPrefix

`public void` **`setPrefix`**`(java.lang.String prefix)`

This method is used to set the prefix value.

setURI

`public void` **`setURI`**`(java.lang.String uri)`

This method is used to get the URI value.

com.objsys.asn1j.runtime
Class Asn1XmlNamespaceCtxt

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XmlNamespaceCtxt
```

public class **Asn1XmlNamespaceCtxt**
extends java.lang.Object

Represents a mapping between namespaces and prefixes. A prefix may map only to a single namespace. A namespace may be bound to multiple prefixes. The default namespace is mapped by DEFAULT_NS_PREFIX. By default, the default namespace is NULL_NS_URI.

Field Summary

static final java.lang.String	DEFAULT_NS_PREFIX Value:
static final java.lang.String	NULL_NS_URI Value:

Constructor Summary

public	Asn1XmlNamespaceCtxt()

Method Summary

java.lang.String	addNamespace(java.lang.String namespaceURI,java.lang.String prefix) Add (or replace) a namespace mapping.
java.lang.String	addUniquePrefix(java.lang.String namespaceURI) Map a unique prefix for the given namespace.
java.lang.String	getExistingPrefix(java.lang.String namespaceURI) Return a prefix for the given namespace URI.
java.lang.String	getNamespaceURI(java.lang.String prefix) Return the namespace URI that the given prefix is mapped to.
java.lang.String	getPrefix(java.lang.String namespaceURI) Return a prefix for the given namespace URI.
boolean	isDefaultNamespaceURI(java.lang.String namespaceURI) Return true if the given namespace is currently the default namespace (ie, DEFAULT_NS_PREFIX is mapped to this URI)

java.util.Set	prefixes() Return an unmodifiable Set of all mapped prefixes

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

DEFAULT_NS_PREFIX

public static final java.lang.String **DEFAULT_NS_PREFIX**

NULL_NS_URI

public static final java.lang.String **NULL_NS_URI**

Constructors

Asn1XmlNamespaceCtxt

public **Asn1XmlNamespaceCtxt**()

Methods

addNamespace

public java.lang.String **addNamespace**(java.lang.String namespaceURI,
java.lang.String prefix)

Add (or replace) a namespace mapping. If prefix was previously mapped, the previously mapped URI is returned. Otherwise, NULL_NS_URI is returned.

Parameters:

namespaceURI
prefix

addUniquePrefix

public java.lang.String **addUniquePrefix**(java.lang.String namespaceURI)

Map a unique prefix for the given namespace.

Returns:

the unique prefix that was generated and mapped to the namespace

getExistingPrefix

public java.lang.String **getExistingPrefix**(java.lang.String namespaceURI)

(continued from last page)

Return a prefix for the given namespace URI. If the DEFAULT_NS_PREFIX is mapped to the URI, it will be returned. Otherwise, if one or more other prefixes are mapped to the URI, one of these will be returned. Finally, if no prefix at all is mapped to the URI, null is returned.

Parameters:

`namespaceURI`

Returns:

getNamespaceURI

`public java.lang.String` **`getNamespaceURI`**`(java.lang.String prefix)`

Return the namespace URI that the given prefix is mapped to. Return NULL_NS_URI if it is not mapped to any URI

Parameters:

`prefix`

getPrefix

`public java.lang.String` **`getPrefix`**`(java.lang.String namespaceURI)`

Return a prefix for the given namespace URI. If one or more prefixes other than DEFAULT_NS_PREFIX is mapped to the URI, one of these will be returned. Otherwise, a generated prefix will be automatically mapped to the URI and returned.

Parameters:

`namespaceURI`

Returns:

isDefaultNamespaceURI

`public boolean` **`isDefaultNamespaceURI`**`(java.lang.String namespaceURI)`

Return true if the given namespace is currently the default namespace (ie, DEFAULT_NS_PREFIX is mapped to this URI)

Parameters:

`namespaceURI`

Returns:

prefixes

`public java.util.Set` **`prefixes`**`()`

Return an unmodifiable Set of all mapped prefixes

Returns:

com.objsys.asn1j.runtime
Class Asn1XmlOutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-com.objsys.asn1j.runtime.Asn1OutputStream
         |
         +-com.objsys.asn1j.runtime.Asn1XmlOutputStream
```

All Implemented Interfaces:
Asn1XmlEncoder, java.io.Flushable, java.io.Closeable

---

public class **Asn1XmlOutputStream**
extends Asn1OutputStream
implements java.io.Closeable, java.io.Flushable, Asn1XmlEncoder

This class implements the output stream to encode ASN.1 messages as specified in the XML Encoding as specified in the XML schema standard. A reference to an object of this type is passed to each of the ASN.1 type encode methods involved in encoding a particular message type.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1OutputStream`
`os`

Constructor Summary

public	`Asn1XmlOutputStream(java.io.OutputStream os)` This constructor creates a buffered XML output stream object with default size of buffer.
public	`Asn1XmlOutputStream(java.io.OutputStream os,int bufSize)` This constructor creates a buffered XML output stream object.

Method Summary

void	`copy(byte value)` This method is used to copy a single byte to the output stream.
void	`copy(byte[] value)` This method copies multiple bytes to the output stream.
void	`copy(byte[] value,int off,int len)` This method copies multiple bytes to the output stream.
void	`copy(java.lang.String value)` This method copies a character string to the output stream.
void	`decrLevel()` This method decrements the element nesting level counter.

void	encodeAttr(java.lang.String qname,java.lang.String value) This method encodes an XML attribute value.
void	encodeBinStrValue(byte[] bits,int nbits) This method encodes XML binary string data
void	encodeData(java.lang.String value) This method encodes XML string data
void	encodeDoubleValue(double value,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an XML REAL (double) value (with start and end tags).
void	encodeEmptyElement(java.lang.String elemName,java.lang.String nsPrefix) This method encodes an XML empty element tag
void	encodeEndDocument() This method encodes standard trailor information at the end of the XML document.
void	encodeEndElement(java.lang.String elemName,java.lang.String nsPrefix) This method encodes an XML end element tag.
void	encodeHexStrValue(byte[] data) This method encodes XML hexadecimal string data
void	encodeNamedValue(java.lang.String valueName,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an XML named value (with start and end tags)
void	encodeNamedValueElement(java.lang.String valueName) This method encodes an XML named value element tag.
void	encodeStartDocument() This method encodes standard header information at the beginning of the XML document.
void	encodeStartElement(java.lang.String elemName,java.lang.String nsPrefix,boolean terminate) This method encodes an XML start element tag with attribute.
void	encodeXSIAttrs() This method encodes XSI attributes.
Asn1XmlEncodeHelper	getHelper() This method returns a reference to the internal helper object.
void	incrLevel() This method increments the element nesting level counter.
void	indent() This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the output stream.
boolean	isCanonical()

void	`setCanonical(boolean value)` This method sets the canonical encoding flag to the given value.
void	`setIndent(int value)` This method sets the number of spaces per indentation.
void	`setMixedContent(boolean value)` This method sets the mixed content flag which indicates that this buffer will be used to encode mixed content.
void	`setTermStart(boolean value)` This method sets the start element termination required flag.
void	`setXSIAttrs(Asn1XmlXSIAttrs value)` This method sets the XSI attributes object to the given value.
void	`write(java.lang.String value)` This method copies a character string to the output stream.

Methods inherited from class `com.objsys.asn1j.runtime.Asn1OutputStream`
`close, flush, getContext, write, write, write, write2Bytes, write4Bytes`

Methods inherited from class `java.io.OutputStream`
`close, flush, write, write, write`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1XmlOutputStream

`public` **`Asn1XmlOutputStream`**`(java.io.OutputStream os)`

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

Parameters:

`os` - The underlying OutputStream object.

Asn1XmlOutputStream

`public` **`Asn1XmlOutputStream`**`(java.io.OutputStream os,`
`int bufSize)`

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

Parameters:

`os` - The underlying OutputStream object.

(continued from last page)

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

Methods

copy

```
public void copy(byte value)
  throws java.io.IOException
```

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

Parameters:

`value` - The byte value to copy

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.

---

copy

```
public void copy(byte[] value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Array of bytes to copy to the output stream

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

copy

```
public void copy(byte[] value,
          int off,
          int len)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

copy

```
public void copy(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method copies a character string to the output stream.

(continued from last page)

Parameters:
`value` - The string value to copy

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

write

```
public void write(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method copies a character string to the output stream.

Parameters:
`value` - The string value to copy

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

decrLevel

```
public void decrLevel()
```

This method decrements the element nesting level counter.

---

encodeData

```
public void encodeData(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML string data

Parameters:
`value` - String value to encode

Throws:
`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeAttr

```
public void encodeAttr(java.lang.String qname,
         java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML attribute value.

Parameters:
`qname` - Attribute qualified name.
`value` - Attribute value in string form.

---

encodeBinStrValue

```
public void encodeBinStrValue(byte[] bits,
         int nbits)
  throws java.io.IOException,
         Asn1Exception
```

(continued from last page)

This method encodes XML binary string data

Parameters:

`bits` - Bit string to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeHexStrValue

```
public void encodeHexStrValue(byte[] data)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML hexadecimal string data

Parameters:

`data` - Data to encode

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeEndDocument

```
public void encodeEndDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeEndElement

```
public void encodeEndElement(java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML end element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeStartDocument

```
public void encodeStartDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeStartElement

```
public void encodeStartElement(java.lang.String elemName,
          java.lang.String nsPrefix,
          boolean terminate)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an XML start element tag with attribute.

Parameters:

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeEmptyElement

```
public void encodeEmptyElement(java.lang.String elemName,
          java.lang.String nsPrefix)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an XML empty element tag

Parameters:

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValueElement

```
public void encodeNamedValueElement(java.lang.String valueName)
  throws Asn1Exception,
          java.io.IOException
```

This method encodes an XML named value element tag.

Parameters:

`valueName` - The name of the element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValue

```
public void encodeNamedValue(java.lang.String valueName,
          java.lang.String elemName,
          java.lang.String nsPrefix)
  throws Asn1Exception,
          java.io.IOException
```

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

Parameters:

`valueName` - The name of value.

(continued from last page)

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeDoubleValue

```
public void encodeDoubleValue(double value,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeXSIAttrs

```
public void encodeXSIAttrs()
  throws Asn1Exception,
         java.io.IOException
```

This method encodes XSI attributes.

---

getHelper

```
public Asn1XmlEncodeHelper getHelper()
```

This method returns a reference to the internal helper object.

Returns:

Reference to mHelper object.

---

incrLevel

```
public void incrLevel()
```

This method increments the element nesting level counter.

---

indent

```
public void indent()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - Any exception thrown by the underlying OutputStream.
`Asn1Exception` - Thrown, if operation is failed.

(continued from last page)

isCanonical

`public boolean isCanonical()`

See Also:
`Asn1XmlXerEncoder`

setCanonical

`public void setCanonical(boolean value)`

This method sets the canonical encoding flag to the given value.

Parameters:
`value` - Canonical encoding flag.

setIndent

`public void setIndent(int value)`

This method sets the number of spaces per indentation. If the value is set to zero, no indentation of the XML data is done.

Parameters:
`value` - Number of spaces per indentation level.

setMixedContent

`public void setMixedContent(boolean value)`

This method sets the mixed content flag which indicates that this buffer will be used to encode mixed content.

Parameters:
`value` - Boolean value

setTermStart

`public void setTermStart(boolean value)`

This method sets the start element termination required flag.

Parameters:
`value` - Boolean value

setXSIAttrs

`public void setXSIAttrs(Asn1XmlXSIAttrs value)`

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

Parameters:
`value` - XSI attributes object

com.objsys.asn1j.runtime
Class Asn1XmlSaxSimpleType

```
java.lang.Object
   ┗-org.xml.sax.helpers.DefaultHandler
      ┗-com.objsys.asn1j.runtime.Asn1XerSaxHandler
         ┗-com.objsys.asn1j.runtime.Asn1XmlSaxSimpleType
```

public class **Asn1XmlSaxSimpleType**
extends Asn1XerSaxHandler

SAX Handler for simple types. This handler captures all text for the first element for which startElement is invoked. Once the all text is captured and the corresponding endElement is invoked, there are two possibilities: 1) if you passed an Asn1Type for element to the constructor, this handler will invoke element.decodeXML to decode the XML text into that object. 2) if you passed null for the constructor's element argument, then you may use getText() to get the XML element's text.

Fields inherited from class `com.objsys.asn1j.runtime.Asn1XerSaxHandler`
`mConsumedStartElement`, `mCurrElemID`, `mCurrState`, `mLevel`, `mStartLevel`, `mXMLElemName`, `XERDATA`, `XEREND`, `XERINIT`, `XERSTART`, `XERUNKNOWN`

Constructor Summary

public	`Asn1XmlSaxSimpleType(Asn1Type element)`

Method Summary

void	`characters(char[] ch,int start,int length)`
void	`endElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName)`
java.lang.String	`getText()` After endElement has been triggered, this may be used to obtain the XML element's text.
void	`startElement(java.lang.String namespaceURI,java.lang.String localName,java.lang.String qName,org.xml.sax.Attributes atts)`

Methods inherited from class `com.objsys.asn1j.runtime.Asn1XerSaxHandler`
`consumeStartElement, endGroup, error, fatalError, getState, init, isComplete, isDecodingAsGroup, matchXMLElemName, setComplete, setLevel, setXMLElemName, warning`

Methods inherited from class `org.xml.sax.helpers.DefaultHandler`
`characters, endDocument, endElement, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator, skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1XmlSaxSimpleType

```
public Asn1XmlSaxSimpleType(Asn1Type element)
```

Methods

startElement

```
public void startElement(java.lang.String namespaceURI,
         java.lang.String localName,
         java.lang.String qName,
         org.xml.sax.Attributes atts)
  throws org.xml.sax.SAXException
```

characters

```
public void characters(char[] ch,
         int start,
         int length)
  throws org.xml.sax.SAXException
```

endElement

```
public void endElement(java.lang.String namespaceURI,
         java.lang.String localName,
         java.lang.String qName)
  throws org.xml.sax.SAXException
```

getText

```
public java.lang.String getText()
```

After endElement has been triggered, this may be used to obtain the XML element's text.

com.objsys.asn1j.runtime
Class Asn1XmlUnexpElemExc

```
java.lang.Object
   ┗-java.lang.Throwable
      ┗-java.lang.Exception
         ┗-java.lang.RuntimeException
            ┗-com.objsys.asn1j.runtime.Asn1Exception
               ┗-com.objsys.asn1j.runtime.Asn1XmlUnexpElemExc
```

public class **Asn1XmlUnexpElemExc**
extends Asn1Exception

This class defines the 'unexpected XML element' exception. This exception is thrown when an element in an XML instance is encountered that is different than what is defined in the schema to occur at that position.

Constructor Summary

public	`Asn1XmlUnexpElemExc(java.lang.String localName)` This constructor creates an exception object with a textual message describing the expected element value.

Methods inherited from class `java.lang.Throwable`

`addSuppressed, fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, getSuppressed, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Asn1XmlUnexpElemExc

`public` **`Asn1XmlUnexpElemExc`**`(java.lang.String localName)`

This constructor creates an exception object with a textual message describing the expected element value.

Parameters:

`localName` - Parsed XML local name

com.objsys.asn1j.runtime
Class Asn1XmlUtil

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XmlUtil
```

public class **Asn1XmlUtil**
extends java.lang.Object

Constructor Summary

public	Asn1XmlUtil()

Method Summary

static void	encodeDouble(Asn1XmlEncoder buffer,double value) This method encodes an ASN.1 real value using the XML encoding (non-XER).
static void	encodeDouble(Asn1XmlEncoder buffer,double value,java.lang.String elemName,java.lang.String nsPrefix) This method encodes an ASN.1 real value using the XML encoding (non-XER).
static java.lang.String	getXMLString(java.lang.String value) This method will convert the given string value into XML character content by escaping special characters in the sting such as ampersand (&), left angle bracket (<), etc..
static boolean	isMinusZero(double value) This method will return true, if value is "minus zero" (-0) special XML value.
static boolean	isNaN(double value) This method will return true, if value is "not-a-number" (NaN) special XML value.
static java.lang.String[]	tokenizeXsdList(java.lang.String listValue) Return an array of strings representing the (lexical) values of an xsd:list

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Asn1XmlUtil

public **Asn1XmlUtil**()

Methods

encodeDouble

```
public static void encodeDouble(Asn1XmlEncoder buffer,
         double value,
         java.lang.String elemName,
         java.lang.String nsPrefix)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

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

---

encodeDouble

```
public static void encodeDouble(Asn1XmlEncoder buffer,
         double value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`buffer` - Encode message buffer object
`value` - Value to be encoded.

---

getXMLString

```
public static java.lang.String getXMLString(java.lang.String value)
```

This method will convert the given string value into XML character content by escaping special characters in the sting such as ampersand (&), left angle bracket (<), etc..

Parameters:

`value` - String to convert

Returns:

Converted string value

---

isMinusZero

```
public static boolean isMinusZero(double value)
```

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

Parameters:

`value` - value to test

Returns:

true, if this value is "-0".

---

isNaN

```
public static boolean isNaN(double value)
```

This method will return true, if value is "not-a-number" (NaN) special XML value.

(continued from last page)

Parameters:

`value` - value to test

Returns:

true, if this value is NaN.

tokenizeXsdList

`public static java.lang.String[] tokenizeXsdList(java.lang.String listValue)`

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

Parameters:

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

Returns:

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

com.objsys.asn1j.runtime
Interface Asn1XmlXerEncoder

All Subinterfaces:
Asn1XmlEncoder, Asn1XerEncoder

---

public interface **Asn1XmlXerEncoder**

This is a base interface for encoding of ASN.1 messages as XML as specified in the W3C standard. It is implemented by both the Asn1XmlEncodeBuffer and Asn1XmlOutputStream.

Method Summary

abstract void	`copy(byte value)` This method is used to copy a single byte to the encode buffer or stream.
abstract void	`copy(byte[] value)` This method copies multiple bytes to the encode buffer or stream.
abstract void	`copy(byte[] value,int off,int len)` This method copies multiple bytes to the encode buffer or stream.
abstract void	`copy(java.lang.String value)` This method copies a character string to the encode buffer or stream.
abstract void	`decrLevel()` This method decrements the element nesting level counter.
abstract void	`encodeBinStrValue(byte[] bits,int nbits)` This method encodes XML binary string data
abstract void	`encodeData(java.lang.String value)` This method encodes XML string data
abstract void	`encodeEndDocument()` This method encodes standard trailor information at the end of the XML document.
abstract void	`encodeHexStrValue(byte[] data)` This method encodes XML hexadecimal string data
abstract void	`encodeNamedValueElement(java.lang.String elemName)` This method encodes an XML named value element tag.
abstract void	`encodeStartDocument()` This method encodes standard header information at the beginning of the XML document.
abstract Asn1Context	`getContext()` Return the context for this encoder.

abstract void	incrLevel() This method increments the element nesting level counter.
abstract void	indent() This methods indents by adding a new-line followed by whitespace corresponding to the current nesting level to the encode buffer.
abstract boolean	isCanonical() Return true if the encoder has been put into canonical mode.

Methods

copy

```
public abstract void copy(byte value)
  throws java.io.IOException
```

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

Parameters:

`value` - The byte value to copy

Throws:

`IOException` - If I/O error occurs.

copy

```
public abstract void copy(byte[] value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Array of bytes to copy to the encode buffer

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

copy

```
public abstract void copy(byte[] value,
          int off,
          int len)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - Array of bytes to copy to the encode buffer
`off` - Starting offset in array
`len` - The length to be encoded

(continued from last page)

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

copy

```
public abstract void copy(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

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

Parameters:

`value` - The string value to copy

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

incrLevel

```
public abstract void incrLevel()
```

This method increments the element nesting level counter.

---

decrLevel

```
public abstract void decrLevel()
```

This method decrements the element nesting level counter.

---

encodeData

```
public abstract void encodeData(java.lang.String value)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML string data

Parameters:

`value` - String value to encode

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

---

encodeBinStrValue

```
public abstract void encodeBinStrValue(byte[] bits,
         int nbits)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML binary string data

Parameters:

`bits` - Bit string to encode

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeHexStrValue

```
public abstract void encodeHexStrValue(byte[] data)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes XML hexadecimal string data

Parameters:

`data` - Data to encode

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeEndDocument

```
public abstract void encodeEndDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeStartDocument

```
public abstract void encodeStartDocument()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

encodeNamedValueElement

```
public abstract void encodeNamedValueElement(java.lang.String elemName)
  throws java.io.IOException,
         Asn1Exception
```

This method encodes an XML named value element tag.

Parameters:

`elemName` - The name of element.

Throws:

`IOException` - If I/O error occurs.
`Asn1Exception` - Thrown, if operation is failed.

getContext

```
public abstract Asn1Context getContext()
```

Return the context for this encoder.

(continued from last page)

indent

```
public abstract void indent()
  throws java.io.IOException,
         Asn1Exception
```

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

Throws:

`IOException` - If I/O error occurs.

`Asn1Exception` - Thrown, if operation is failed.

---

isCanonical

```
public abstract boolean isCanonical()
```

Return true if the encoder has been put into canonical mode. For XER, canonical mode turns off extra whitespace output. For XML and XER, canonical mode signals the generated encoders to always encode ASN.1 elements that have a DEFAULT value.

com.objsys.asn1j.runtime
Class Asn1XmlXSIAttrs

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Asn1XmlXSIAttrs
```

public class **Asn1XmlXSIAttrs**
extends java.lang.Object

Constructor Summary

public	`Asn1XmlXSIAttrs()`

Method Summary

void	`encode(Asn1XmlEncoder buffer)` This method writes the set of XSI attributes and the XSI namespace declaration to the encode buffer or stream.
java.lang.String	`getNoNSSchemaLoc()` This method gets the current XSI no namespace schema location attribute setting.
java.lang.String	`getSchemaLocation()` This method gets the current XSI schema location attribute setting.
java.lang.String	`getXSIType()` This method gets the current XSI type setting.
void	`setNoNSSchemaLoc(java.lang.String value)` This method sets the current XSI no namespace schema location attribute setting.
void	`setSchemaLocation(java.lang.String value)` This method sets the current XSI schema location attribute setting.
void	`setXSIType(java.lang.String value)` This method sets the current XSI type setting.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

(continued from last page)

Asn1XmlXSIAttrs

`public` **`Asn1XmlXSIAttrs`**`()`

Methods

getSchemaLocation

`public java.lang.String` **`getSchemaLocation`**`()`

This method gets the current XSI schema location attribute setting.

Returns:

XSI schema location attribute value.

---

setSchemaLocation

`public void` **`setSchemaLocation`**`(java.lang.String value)`

This method sets the current XSI schema location attribute setting.

Parameters:

`value` - XSI schema location attribute value.

---

getNoNSSchemaLoc

`public java.lang.String` **`getNoNSSchemaLoc`**`()`

This method gets the current XSI no namespace schema location attribute setting.

Returns:

No namespace XSI schema location attribute value.

---

setNoNSSchemaLoc

`public void` **`setNoNSSchemaLoc`**`(java.lang.String value)`

This method sets the current XSI no namespace schema location attribute setting.

Parameters:

`value` - No namespace XSI schema location attribute value.

---

getXSIType

`public java.lang.String` **`getXSIType`**`()`

This method gets the current XSI type setting.

Returns:

XSI type attribute value.

---

setXSIType

`public void` **`setXSIType`**`(java.lang.String value)`

This method sets the current XSI type setting.

Parameters:

`value` - XSI type attribute value.

encode

```
public void encode(Asn1XmlEncoder buffer)
  throws Asn1Exception,
         java.io.IOException
```

This method writes the set of XSI attributes and the XSI namespace declaration to the encode buffer or stream.

Parameters:

`buffer` - XML encode buffer or stream object.

com.objsys.asn1j.runtime
Class Base64

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Base64
```

public class **Base64**
extends java.lang.Object

Field Summary

static final int	DECODE Specify decoding. Value: **0**
static final int	DONT_BREAK_LINES Don't break lines when encoding (violates strict Base64 specification) Value: **8**
static final int	ENCODE Specify encoding. Value: **1**
static final int	GZIP Specify that data should be gzip-compressed. Value: **2**
static final int	NO_OPTIONS No options specified. Value: **0**

Method Summary

static byte[]	decode(byte[] source,int off,int len) Very low-level access to decoding ASCII characters in the form of a byte array.
static byte[]	decode(java.lang.String s) Decodes data from Base64 notation, automatically detecting gzip-compressed data and decompressing it.
static byte[]	decodeFromFile(java.lang.String filename) Convenience method for reading a base64-encoded file and decoding it.
static boolean	decodeToFile(java.lang.String dataToDecode,java.lang.String filename) Convenience method for decoding data to a file.

static java.lang.Object	decodeToObject(java.lang.String encodedObject) Attempts to decode Base64 data and deserialize a Java Object within.
static java.lang.String	encodeBytes(byte[] source) Encodes a byte array into Base64 notation.
static java.lang.String	encodeBytes(byte[] source,int options) Encodes a byte array into Base64 notation.
static java.lang.String	encodeBytes(byte[] source,int off,int len) Encodes a byte array into Base64 notation.
static java.lang.String	encodeBytes(byte[] source,int off,int len,int options) Encodes a byte array into Base64 notation.
static java.lang.String	encodeFromFile(java.lang.String filename) Convenience method for reading a binary file and base64-encoding it.
static java.lang.String	encodeObject(java.io.Serializable serializableObject) Serializes an object and returns the Base64-encoded version of that serialized object.
static java.lang.String	encodeObject(java.io.Serializable serializableObject,int options) Serializes an object and returns the Base64-encoded version of that serialized object.
static boolean	encodeToFile(byte[] dataToEncode,java.lang.String filename) Convenience method for encoding data to a file.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

NO_OPTIONS

public static final int **NO_OPTIONS**

No options specified. Value is zero.

ENCODE

public static final int **ENCODE**

Specify encoding.

DECODE

public static final int **DECODE**

Specify decoding.

(continued from last page)

GZIP

```
public static final int GZIP
```

Specify that data should be gzip-compressed.

DONT_BREAK_LINES

```
public static final int DONT_BREAK_LINES
```

Don't break lines when encoding (violates strict Base64 specification)

Methods

encodeObject

```
public static java.lang.String encodeObject(java.io.Serializable serializableObject)
```

Serializes an object and returns the Base64-encoded version of that serialized object. If the object cannot be serialized or there is another error, the method will return null. The object is not GZip-compressed before being encoded.

Parameters:

`serializableObject` - The object to encode

Returns:

The Base64-encoded object

Since:

1.4

encodeObject

```
public static java.lang.String encodeObject(java.io.Serializable serializableObject,
        int options)
```

Serializes an object and returns the Base64-encoded version of that serialized object. If the object cannot be serialized or there is another error, the method will return null. Valid options:

```
GZIP: gzip-compresses object before encoding it.
   DONT_BREAK_LINES: don't break lines at 76 characters
     Note: Technically, this makes your encoding non-compliant.
```

Example: `encodeObject( myObj, Base64.GZIP )` or Example: `encodeObject( myObj, Base64.GZIP | Base64.DONT_BREAK_LINES )`

Parameters:

`serializableObject` - The object to encode

`options` - Specified options

Returns:

The Base64-encoded object

See Also:

`Base64.GZIP()`

`Base64.DONT_BREAK_LINES()`

Since:

2.0

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source)
```

Encodes a byte array into Base64 notation. Does not GZip-compress data.

(continued from last page)

Parameters:

`source` - The data to convert

Since:

1.4

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source,
                                           int options)
```

Encodes a byte array into Base64 notation. Valid options: `GZIP: gzip-compresses object before encoding it.`
`DONT_BREAK_LINES: don't break lines at 76 characters`
`Note: Technically, this makes your encoding non-compliant.`
Example: `encodeBytes( myData, Base64.GZIP )`or Example: `encodeBytes( myData, Base64.GZIP | Base64.DONT_BREAK_LINES )`

Parameters:

`source` - The data to convert
`options` - Specified options

See Also:

`Base64.GZIP()`
`Base64.DONT_BREAK_LINES()`

Since:

2.0

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source,
                                           int off,
                                           int len)
```

Encodes a byte array into Base64 notation. Does not GZip-compress data.

Parameters:

`source` - The data to convert
`off` - Offset in array where conversion should begin
`len` - Length of data to convert

Since:

1.4

---

encodeBytes

```
public static java.lang.String encodeBytes(byte[] source,
                                           int off,
                                           int len,
                                           int options)
```

Encodes a byte array into Base64 notation. Valid options: `GZIP: gzip-compresses object before encoding it.`
`DONT_BREAK_LINES: don't break lines at 76 characters`
`Note: Technically, this makes your encoding non-compliant.`
Example: `encodeBytes( myData, Base64.GZIP )`or Example: `encodeBytes( myData, Base64.GZIP | Base64.DONT_BREAK_LINES )`

Parameters:

`source` - The data to convert
`off` - Offset in array where conversion should begin
`len` - Length of data to convert
`options` - Specified options

(continued from last page)

See Also:

`Base64.GZIP()`
`Base64.DONT_BREAK_LINES()`

Since:

2.0

---

decode

```
public static byte[] decode(byte[] source,
                            int off,
                            int len)
```

Very low-level access to decoding ASCII characters in the form of a byte array. Does not support automatically gunzipping or any other "fancy" features.

Parameters:

`source` - The Base64 encoded data
`off` - The offset of where to begin decoding
`len` - The length of characters to decode

Returns:

decoded data

Since:

1.3

---

decode

```
public static byte[] decode(java.lang.String s)
```

Decodes data from Base64 notation, automatically detecting gzip-compressed data and decompressing it.

Parameters:

`s` - the string to decode

Returns:

the decoded data

Since:

1.4

---

decodeToObject

```
public static java.lang.Object decodeToObject(java.lang.String encodedObject)
```

Attempts to decode Base64 data and deserialize a Java Object within. Returns nullif there was an error.

Parameters:

`encodedObject` - The Base64 data to decode

Returns:

The decoded and deserialized object

Since:

1.5

(continued from last page)

encodeToFile

```
public static boolean encodeToFile(byte[] dataToEncode,
          java.lang.String filename)
```

Convenience method for encoding data to a file.

Parameters:

`dataToEncode` - byte array of data to encode in base64 form
`filename` - Filename for saving encoded data

Returns:

trueif successful, falseotherwise

Since:

2.1

---

decodeToFile

```
public static boolean decodeToFile(java.lang.String dataToDecode,
          java.lang.String filename)
```

Convenience method for decoding data to a file.

Parameters:

`dataToDecode` - Base64-encoded data as a string
`filename` - Filename for saving decoded data

Returns:

trueif successful, falseotherwise

Since:

2.1

---

decodeFromFile

```
public static byte[] decodeFromFile(java.lang.String filename)
```

Convenience method for reading a base64-encoded file and decoding it.

Parameters:

`filename` - Filename for reading encoded data

Returns:

decoded byte array or null if unsuccessful

Since:

2.1

---

encodeFromFile

```
public static java.lang.String encodeFromFile(java.lang.String filename)
```

Convenience method for reading a binary file and base64-encoding it.

Parameters:

`filename` - Filename for reading binary data

Returns:

base64-encoded string or null if unsuccessful

(continued from last page)

Since:

2.1

com.objsys.asn1j.runtime
Class Base64.InputStream

```
java.lang.Object
   |
   +-java.io.InputStream
      |
      +-java.io.FilterInputStream
         |
         +-com.objsys.asn1j.runtime.Base64.InputStream
```

public static class **Base64.InputStream**
extends java.io.FilterInputStream

A `Base64.InputStream` will read data from another java.io.InputStream, given in the constructor, and encode/decode to/from Base64 notation on the fly.

See Also:
Base64

Fields inherited from class `java.io.FilterInputStream`
`in`

Constructor Summary

public	`InputStream(java.io.InputStream in)` Constructs a `Base64.InputStream` in DECODE mode.
public	`InputStream(java.io.InputStream in,int options)` Constructs a `Base64.InputStream` in either ENCODE or DECODE mode.

Method Summary

int	`read()` Reads enough of the input stream to convert to/from Base64 and returns the next byte.
int	`read(byte[] dest,int off,int len)` Calls `#read()` repeatedly until the end of stream is reached or lenbytes are read.

Methods inherited from class `java.io.FilterInputStream`
`available, close, mark, markSupported, read, read, read, reset, skip`

Methods inherited from class `java.io.InputStream`
`available, close, mark, markSupported, read, read, read, reset, skip`

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

InputStream

```
public InputStream(java.io.InputStream in)
```

Constructs a Base64.InputStream in DECODE mode.

Parameters:

in - the java.io.InputStreamfrom which to read data.

Since:

1.3

InputStream

```
public InputStream(java.io.InputStream in,
                   int options)
```

Constructs a Base64.InputStream in either ENCODE or DECODE mode. Valid options: ENCODE or DECODE: Encode or Decode as data is read.
DONT_BREAK_LINES: don't break lines at 76 characters
(only meaningful when encoding)
Note: Technically, this makes your encoding non-compliant.
Example: new Base64.InputStream(in, Base64.DECODE)

Parameters:

in - the java.io.InputStreamfrom which to read data.
options - Specified options

See Also:

Base64.ENCODE()
Base64.DECODE()
Base64.DONT_BREAK_LINES()

Since:

2.0

Methods

read

```
public int read()
  throws java.io.IOException
```

Reads enough of the input stream to convert to/from Base64 and returns the next byte.

Returns:

next byte

Since:

1.3

(continued from last page)

read

```
public int read(byte[] dest,
                int off,
                int len)
  throws java.io.IOException
```

Calls #`read()` repeatedly until the end of stream is reached or lenbytes are read. Returns number of bytes read into array or -1 if end of stream is encountered.

Parameters:

`dest` - array to hold values
`off` - offset for array
`len` - max number of bytes to read into array

Returns:

bytes read into array or -1 if end of stream is encountered.

Since:

1.3

com.objsys.asn1j.runtime
Class Base64.OutputStream

```
java.lang.Object
   |
   +-java.io.OutputStream
      |
      +-java.io.FilterOutputStream
         |
         +-com.objsys.asn1j.runtime.Base64.OutputStream
```

public static class **Base64.OutputStream**
extends java.io.FilterOutputStream

A `Base64.OutputStream` will write data to another java.io.OutputStream, given in the constructor, and encode/decode to/from Base64 notation on the fly.

See Also:
Base64

Fields inherited from class `java.io.FilterOutputStream`
`out`

Constructor Summary

`public`	`OutputStream(java.io.OutputStream out)` Constructs a `Base64.OutputStream` in ENCODE mode.
`public`	`OutputStream(java.io.OutputStream out,int options)` Constructs a `Base64.OutputStream` in either ENCODE or DECODE mode.

Method Summary

`void`	`close()` Flushes and closes (I think, in the superclass) the stream.
`void`	`flushBase64()` Method added by PHIL.
`void`	`resumeEncoding()` Resumes encoding of the stream.
`void`	`suspendEncoding()` Suspends encoding of the stream.
`void`	`write(byte[] theBytes,int off,int len)` Calls #`write(int)` repeatedly until lenbytes are written.
`void`	`write(int theByte)` Writes the byte to the output stream after converting to/from Base64 notation.

Methods inherited from class `java.io.FilterOutputStream`
`close, flush, write, write, write`

Methods inherited from class `java.io.OutputStream`
`close, flush, write, write, write`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

OutputStream

`public` **`OutputStream`**`(java.io.OutputStream out)`

Constructs a `Base64.OutputStream` in ENCODE mode.

Parameters:

`out` - the java.io.OutputStreamto which data will be written.

Since:

1.3

OutputStream

`public` **`OutputStream`**`(java.io.OutputStream out,`
`int options)`

Constructs a `Base64.OutputStream` in either ENCODE or DECODE mode. Valid options: `ENCODE or DECODE: Encode or Decode as data is read.`
`DONT_BREAK_LINES: don't break lines at 76 characters`
`(only meaningful when encoding)`
`Note: Technically, this makes your encoding non-compliant.`
Example: `new Base64.OutputStream( out, Base64.ENCODE )`

Parameters:

`out` - the java.io.OutputStreamto which data will be written.
`options` - Specified options.

See Also:

`Base64.ENCODE()`
`Base64.DECODE()`
`Base64.DONT_BREAK_LINES()`

Since:

1.3

Methods

(continued from last page)

write

```
public void write(int theByte)
  throws java.io.IOException
```

Writes the byte to the output stream after converting to/from Base64 notation. When encoding, bytes are buffered three at a time before the output stream actually gets a write() call. When decoding, bytes are buffered four at a time.

Parameters:

`theByte` - the byte to write

Since:

1.3

---

write

```
public void write(byte[] theBytes,
         int off,
         int len)
  throws java.io.IOException
```

Calls #`write(int)` repeatedly until lenbytes are written.

Parameters:

`theBytes` - array from which to read bytes
`off` - offset for array
`len` - max number of bytes to read into array

Since:

1.3

---

flushBase64

```
public void flushBase64()
  throws java.io.IOException
```

Method added by PHIL. [Thanks, PHIL. -Rob] This pads the buffer without closing the stream.

---

close

```
public void close()
  throws java.io.IOException
```

Flushes and closes (I think, in the superclass) the stream.

Since:

1.3

---

suspendEncoding

```
public void suspendEncoding()
  throws java.io.IOException
```

Suspends encoding of the stream. May be helpful if you need to embed a piece of base640-encoded data in a stream.

Since:

1.5.1

---

resumeEncoding

```
public void resumeEncoding()
```

Resumes encoding of the stream. May be helpful if you need to embed a piece of base640-encoded data in a stream.

(continued from last page)

Since:

1.5.1

com.objsys.asn1j.runtime
Class BooleanHolder

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.BooleanHolder
```

public final class **BooleanHolder**
extends java.lang.Object

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.value`.

Field Summary

boolean	value

Constructor Summary

public	BooleanHolder()
public	BooleanHolder(boolean initial)

Methods inherited from class `java.lang.Object`
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

value

```
public boolean value
```

Constructors

BooleanHolder

```
public BooleanHolder()
```

(continued from last page)

BooleanHolder

```
public BooleanHolder(boolean initial)
```

com.objsys.asn1j.runtime
Class Diag

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.Diag
```

public class **Diag**
extends java.lang.Object

This class is used for printing diagnostic messages for debugging the run-time components. It allows messages to be easily switched on and off.

Method Summary

static void	hexDump(byte[] bytes)
static void	hexDump(byte[] bytes,int traceLevel)
static void	hexDump(java.io.InputStream in,java.io.PrintStream out)
static Diag	instance()
boolean	isEnabled()
boolean	isEnabled(int traceLevel)
void	println(java.lang.String s)
void	println(java.lang.String s,int traceLevel)
static void	prtln(byte[] b,int offset,int nbytes)
static void	prtln(byte[] b,int offset,int nbytes,int tl)
static void	prtln(java.lang.String s)
static void	prtln(java.lang.String s,int traceLevel)
boolean	setEnabled(boolean value)
void	setPrintStream(java.io.PrintStream ps)
static int	setTraceLevel(int level)
int	setTraceLevel2(int level)

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods

isEnabled

```
public boolean isEnabled()
```

isEnabled

```
public boolean isEnabled(int traceLevel)
```

hexDump

```
public static void hexDump(byte[] bytes)
```

hexDump

```
public static void hexDump(byte[] bytes,
         int traceLevel)
```

hexDump

```
public static void hexDump(java.io.InputStream in,
         java.io.PrintStream out)
```

prtln

```
public static void prtln(java.lang.String s)
```

prtln

```
public static void prtln(java.lang.String s,
         int traceLevel)
```

prtln

```
public static void prtln(byte[] b,
         int offset,
         int nbytes,
         int tl)
```

prtln

```
public static void prtln(byte[] b,
          int offset,
          int nbytes)
```

println

```
public void println(java.lang.String s)
```

println

```
public void println(java.lang.String s,
          int traceLevel)
```

setEnabled

```
public boolean setEnabled(boolean value)
```

setPrintStream

```
public void setPrintStream(java.io.PrintStream ps)
```

setTraceLevel2

```
public int setTraceLevel2(int level)
```

setTraceLevel

```
public static int setTraceLevel(int level)
```

instance

```
public static Diag instance()
```

com.objsys.asn1j.runtime
Class IntHolder

```
java.lang.Object
   |
   +-com.objsys.asn1j.runtime.IntHolder
```

public final class **IntHolder**
extends java.lang.Object

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.value`.

Field Summary

int	value

Constructor Summary

public	IntHolder()
public	IntHolder(int initial)

Methods inherited from class `java.lang.Object`
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Fields

value

`public int` **`value`**

Constructors

IntHolder

`public` **`IntHolder`**`()`

(continued from last page)

IntHolder

```
public IntHolder(int initial)
```