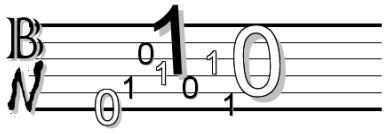
BinaryNotes

Developers Guide



http://bnotes.sourceforge.net

Table of Contents

e-mail : abdulla.abdurakhmanov@gmail.com

1. Overview

BinaryNotes is the Open Source ASN.1 (Abstract Syntax Notation One) framework for Java and .NET.

In telecommunications and computer networking, Abstract Syntax Notation One (ASN.1) is a standard and flexible notation that describes data structures for representing, encoding, transmitting, and decoding data. It provides a set of formal rules for describing the structure of objects that are independent of machine-specific encoding techniques and is a precise, formal notation that removes ambiguities.

ASN.1 is a joint ISO and ITU-T standard, originally defined in 1984 as part of CCITT X.409:1984. ASN.1 moved to its own standard, X.208, in 1988 due to wide applicability. The substantially revised 1995 version is covered by the X.680 series.

ASN.1 defines the abstract syntax of information but does not restrict the way the information is encoded. Various ASN.1 encoding rules provide the transfer syntax (a concrete representation) of the data values whose abstract syntax is described in ASN.1.

The standard ASN.1 encoding rules include[1]:

* Basic Encoding Rules (BER)
* Canonical Encoding Rules (CER)
* Distinguished Encoding Rules (DER)
* XML Encoding Rules (XER) and Extended XML Encoding Rules (EXER)
* Packed Encoding Rules (PER)
* Generic String Encoding Rules (GSER)

ASN.1 together with specific ASN.1 encoding rules facilitates the exchange of structured data especially between application programs over networks by describing data structures in a way that is independent of machine architecture and implementation language.

Application layer protocols such as X.400 electronic mail, X.500 and LDAP directory services, H.323 (VoIP) and SNMP use ASN.1 to describe the PDUs they exchange. It is also extensively used in the Access and Non-Access Strata of UMTS. There are many other application domains of ASN.1

Materials from http://en.wikipedia.org/wiki/ASN.1 are used for writing this section. For more details please go to at this reference.

[1] CER, XER/EXER, GSER is not supported by BinaryNotes.

2. Introduction

The framework contains:

- Encoding/decoding library. The library has BER (Basic Encoding Rules), DER and PER (Packet Encoding Rules) implementation.
- BNCompiler - the extensible ASN.1 compiler which is able to generate Java or C# code for the specified ASN.1 input file. The generated code has annotations/attributes that uses the compiler in runtime. You can customize the generated files by change the original XSL-templates or create your own templates.
- Message Queues – the own MQ implementation based on ASN.1 encoding.

3. Requirements

BinaryNotes required:

- Java Platform Standard Edition v1.5 or newer (.NET developer may use JRE instead JDK[2]).
- .NET 2.0 (Only for .NET developers) or newer

[2] .NET developer needs JVM too. The compiler run only under JavaSE and has not implementation for .NET. But the runtime library is native for .NET

e-mail : abdulla.abdurakhmanov@gmail.com

4. BinaryNotes Compiler

BinaryNotes Compiler (BNCompiler) is tool for generating the class/method declarations for specified ASN.1 specification (input file). The compiler generates the class declarations for ready to use with by runtime BinaryNotes library. But the developer may not use the compiler and implement own classes but usually this is more difficult.

4.1. Generating process

The generated classes has annotations/attributes and simple properties which uses the runtime library. For Java it's JavaBean properties, for .NET using native C# language properties[3].

For example using the following simple ASN.1 declaration:

```
FOOBAR
DEFINITIONS IMPLICIT TAGS ::= BEGIN
  TestSequence ::= SEQUENCE {
        field1 INTEGER,
        field2 PrintableString,
        field3 UTF8String OPTIONAL,
        field4 BOOLEAN DEFAULT false,
        field4 CHOICE {
                field1    INTEGER,
                field2    REAL
        }
  }
END
```

BNCompiler generates for Java:

```
package foobar;
//
// This file was generated by the BinaryNotes compiler.
// See http://bnotes.sourceforge.net
// Any modifications to this file will be lost upon recompilation of the source ASN.1.
//

import org.bn.*;
import org.bn.annotations.*;
import org.bn.annotations.constraints.*;
import org.bn.coders.*;
import org.bn.types.*;

   @ASN1Sequence ( name = "TestSequence", isSet = false )
   public class TestSequence {

         @ASN1Integer( name = "" )
         @ASN1Element ( name = "field1", isOptional =  false , hasTag =  false  , hasDefaultValue =  false  )
         private Long field1 = null;

         @ASN1String( name = "", stringType =  UniversalTag.PrintableString , isUCS = false )
         @ASN1Element ( name = "field2", isOptional =  false , hasTag =  false  , hasDefaultValue =  false  )
         private String field2 = null;

         @ASN1String( name = "", stringType = UniversalTag.UTF8String , isUCS = false )
         @ASN1Element ( name = "field3", isOptional =  true , hasTag =  false  , hasDefaultValue =  false  )
        private String field3 = null;

         @ASN1Boolean( name = "" )
         @ASN1Element ( name = "field4", isOptional =  false , hasTag =  false  , hasDefaultValue =  true  )
        private Boolean field4 = null;
```

[3] This documentation doesn't describe annotation using declaration now. Maybe in later version this will be fixed.

e-mail : abdulla.abdurakhmanov@gmail.com

```
	@ASN1Choice ( name = "field4" )
	public class Field4ChoiceType {
		@ASN1Integer( name = "" )
		@ASN1Element ( name = "field1", isOptional =  false , hasTag =  false  , hasDefaultValue =  false  )
		private Long field1 = null;

		@ASN1Real( name = "" )
		@ASN1Element ( name = "field2", isOptional =  false , hasTag =  false  , hasDefaultValue =  false  )
		private Double field2 = null;

		public Long getField1 () {
			return this.field1;
	}

	public boolean isField1Selected () {
			return this.field1 != null;
	}

	private void setField1 (Long value) {
			this.field1 = value;
	}

	public void selectField1 (Long value) {
			this.field1 = value;
		setField2(null);
	}

	public Double getField2 () {
			return this.field2;
	}

	public boolean isField2Selected () {
			return this.field2 != null;
	}

	private void setField2 (Double value) {
			this.field2 = value;
	}

	public void selectField2 (Double value) {
			this.field2 = value;
			setField1(null);
	}
}

	@ASN1Element ( name = "field4", isOptional =  false , hasTag =  false  , hasDefaultValue =  false  )
	 private Field4ChoiceType field4 = null;

public Long getField1 () {
   return this.field1;
}

public void setField1 (Long value) {
   this.field1 = value;
}

public String getField2 () {
   return this.field2;
}

public void setField2 (String value) {
   this.field2 = value;
}

public String getField3 () {
   return this.field3;
}

public boolean isField3Present () {
   return this.field3 == null;
}
```

```
      public void setField3 (String value) {
         this.field3 = value;
      }

      public Boolean getField4 () {
         return this.field4;
      }

      public void setField4 (Boolean value) {
         this.field4 = value;
      }

      public Field4ChoiceType getField4 () {
         return this.field4;
      }

      public void setField4 (Field4ChoiceType value) {
         this.field4 = value;
      }

      public void initWithDefaults() {
         Boolean param_Field4 = new Boolean (false);
         setField4(param_Field4);
      }

   }
```

And BNCompiler generates for C# (.Net):

```
//
// This file was generated by the BinaryNotes compiler.
// See http://bnotes.sourceforge.net
// Any modifications to this file will be lost upon recompilation of the source ASN.1.
//

using System;
using org.bn.attributes;
using org.bn.attributes.constraints;
using org.bn.coders;
using org.bn.types;

namespace foobar {


   [ASN1Sequence ( Name = "TestSequence", IsSet = false  )]
   public class TestSequence {

        private long field1_ ;
        [ASN1Integer( Name = "" )]
        [ASN1Element ( Name = "field1", IsOptional =  false , HasTag =  false  , HasDefaultValue =  false )  ]
        public long Field1
        {
                        get { return field1_; }
                        set { field1_ = value;  }
        }

        private string field2_ ;
        [ASN1String( Name = "", StringType =  UniversalTags.PrintableString , IsUCS = false )]
        [ASN1Element ( Name = "field2", IsOptional =  false , HasTag =  false  , HasDefaultValue =  false )  ]
        public string Field2
        {
                        get { return field2_; }
                        set { field2_ = value;  }
        }

        private string field3_ ;
        private bool  field3_present = false ;
        [ASN1String( Name = "", StringType =  UniversalTags.UTF8String , IsUCS = false )]
        [ASN1Element ( Name = "field3", IsOptional =  true , HasTag =  false  , HasDefaultValue =  false )  ]
        public string Field3
        {
```

```
                        get { return field3_; }
                        set { field3_ = value; field3_present = true;  }
        }

        private bool field4_ ;
        [ASN1Boolean( Name = "" )]
        [ASN1Element ( Name = "field4", IsOptional =  false , HasTag =  false  , HasDefaultValue =  true )  ]
        public bool Field4
        {
                        get { return field4_; }
                        set { field4_ = value;  }
        }

        private Field4ChoiceType field4_ ;
        [ASN1Choice ( Name = "field4" )]
        public class Field4ChoiceType {

                private long field1_ ;
                private bool  field1_selected = false ;

                [ASN1Element ( Name = "field1", IsOptional =  false , HasTag =  false  , HasDefaultValue =  false )  ]
                [ASN1Integer( Name = "" )]
                public long Field1
                {
                                get { return field1_; }
                                set { selectField1(value); }
                }

                private double field2_ ;
                private bool  field2_selected = false ;

                [ASN1Element ( Name = "field2", IsOptional =  false , HasTag =  false  , HasDefaultValue =  false )  ]
                [ASN1Real( Name = "" )]
                public double Field2
                {
                                get { return field2_; }
                                set { selectField2(value); }
                }

                public bool isField1Selected () {
                                return this.field1_selected ;
                }

                public void selectField1 (long val) {
                                this.field1_ = val;
                                this.field1_selected = true;
                        this.field2_selected = false;
                }

                public bool isField2Selected () {
                                return this.field2_selected ;
                }

                public void selectField2 (double val) {
                                this.field2_ = val;
                                this.field2_selected = true;
                        this.field1_selected = false;
                }
        }

      [ASN1Element ( Name = "field4", IsOptional =  false , HasTag =  false  , HasDefaultValue =  false )  ]
      public Field4ChoiceType Field4
      {
         get { return field4_; }
         set { field4_ = value;  }
      }

      public bool isField3Present () {
          return this.field3_present == true;
      }

      public void initWithDefaults() {
```

```
        bool param_Field4 = false;
        Field4 = param_Field4;
      }
    }
}
```

And this class (classes) can be used for in your code as usually. For Java:

```
TestSequence sequence = new TestSequence();
sequence.setField1(10L);
sequence.setField3("Hello");
// Inner class for implicity ASN.1 type declaration
TestSequence.Field4ChoiceType choice = sequence.new Field4ChoiceType();
choice.selectField2(0.5);
sequence.setField4(choice);
```

For C#:

```
TestSequence sequence = new TestSequence();
sequence.Field1  = 10L;
sequence.Field3 = "Hello";
// Inner class for implicity ASN.1 type declaration
TestSequence.Field4ChoiceType choice = TestSequence.Field4ChoiceType ();
choice.selectField2(0.5);
sequence.Field4 = choice;
```

4.2. BNCompiler command line options

BNCompiler can be executed by bncompiler.cmd script (Win32) or may be fork from ANT-tool (Compiler main class is *org.bn.compiler.Main*).

The compiler is processing the following command line options:

Long option name	Short name	Mandatory	Description	Example
--file	-f	Yes	The source input ASN.1 file	-f mytest.asn
--moduleName	-m	Yes	The translate module name (must be available directory in modules path)	-m java -m cs
--modulesPath	-mp	No	Path to modules directory which contains XSL templates for translating. Default is current directory + "modules/"	-mp d:\modules
--outputDir	-o	No	Output path for generating files. Default is current directory + "output/"	-o org/my/superpackage
--namespace	-ns	No	Namespace/Package name for generated files. Default is ASN.1 module name.	-ns org.my.superpackage

Example of use compiler for C# (Win32):

```
D:\BinaryNotes\Dist\bin\bncompiler.cmd –m cs –o test/org/company –ns test.org.company –f test.asn
```

Example of use complier with ANT-tool:

```
  <path id="library.BN">
    <pathelement location="${dist.path.lib}/binarynotes.jar"/>
  </path>
  <path id="classpath">
    <path refid="library.BN"/>
  </path>
  <path id="bndepends.path">
    <pathelement path="classes"/>
    <pathelement path="${depends.libs.path}/lineargs.jar"/>
    <pathelement path="${depends.libs.path}/antlr.jar"/>
    <pathelement path="${depends.libs.path}/activation.jar"/>
    <pathelement path="${depends.libs.path}/jaxb-api.jar"/>
    <pathelement path="${depends.libs.path}/jaxb-impl.jar"/>
    <pathelement path="${depends.libs.path}/jaxb1-impl.jar"/>
```

```
  <pathelement path="${depends.libs.path}/jsr173_1.0_api.jar"/>
  <pathelement path="${dist.path.lib}/java/binarynotes.jar"/>
  <pathelement path="${dist.path.bin}/bncompiler.jar"/>
 </path>
 <target name="bncompile" depends="init">
  <java classname="org.bn.compiler.Main" fork="true">
   <classpath refid="bndepends.path"/>
   <arg value="-mp"/>
   <arg value="${dist.path.bin}/modules"/>
   <arg value="-m"/>
   <arg value="java"/>
   <arg value="-o"/>
   <arg value="src/org/bn/mq/protocol"/>
   <arg value="-ns"/>
   <arg value="org.bn.mq.protocol"/>
   <arg value="-f"/>
   <arg value="../asn/test.asn"/>
  </java>
 </target>
```

4.3. Extending/Customization of BNCompiler

The compiler distributed with predefined templates (translation modules) for C#/Java. But you can create your own translation modules.
Translation modules is XSL-scripts with predefined structure. BNCompiler depending on moduleName command line option execute some translation module.
Warning: The predetermined modules periodically changes, and if you want customize please create own module and to not change standard modules, but not forget about bugfix updates (may use with diff tools).

e-mail : abdulla.abdurakhmanov@gmail.com

5. BinaryNotes Library

The library supports various encodings standards.
The version 1.3 supports:

- BER
- DER
- PER (Aligned/Unaligned)

The factory for creating Encoder/Decoder implementation is *org.bn.CoderFactory.* An encoder interface is defined as *org.bn.IEncoder,* and decoder as *org.bn.IDecoder.*

CoderFactory is Singleton[4] and can create Encoder/Decoder by specified encoding schema name:

- "BER" for BER encoding
- "DER" for DER encoding
- "PER" or "PER/Aligned" or "PER/A" for PER Aligned encoding
- "PER/Unaligned" or "PER/U" for PER Unaligned encoding

The following code describes creating encoder and decoder.
For Java:

```
// Encoder for Java
IEncoder<DataSeq> encoder = CoderFactory.getInstance().newEncoder("BER");

// Decoder for Java
IDecoder decoder = CoderFactory.getInstance().newDecoder("BER");
```

For C#:

```
// Encoder for C#
IEncoder encoder = CoderFactory.getInstance().newEncoder("BER");

// Decoder for C#
IDecoder decoder = CoderFactory.getInstance().newDecoder("BER");
```

IEncoder contain primary method <T> encode(T obj, OutputStream stream), and IDecoder contain primary method decode<T>(InputStream stream, Class<T> objClass).

Java example:

```
...
// Encoding for Java
TestSequence sequence = new TestSequence();
sequence.setField1(10L);
sequence.setField3("Hello");
// Inner class for implicity ASN.1 type declaration
TestSequence.Field4ChoiceType choice = sequence.new Field4ChoiceType();
choice.selectField2(0.5);
sequence.setField4(choice);
IEncoder< TestSequence> encoder = CoderFactory.getInstance().newEncoder("BER");
ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
encoder.encode(sequence, outputStream);

...
// Decoding for Java
IDecoder decoder = CoderFactory.getInstance().newDecoder("BER");
TestSequence seq = decoder.decode(stream, TestSequence.class);
System.out.println(seq.getField1());
if(seq.isField3Present())
```

[4] From the GoF (Gang-Of-Four) Design Pattern Book definitions

```
        System.out.println(seq.getField3());
...
```

C# example:

```
...
// Encoding for C#
TestSequence sequence = new TestSequence();
sequence.Field1  = 10L;
sequence.Field3 = "Hello";
// Inner class for implicity ASN.1 type declaration
TestSequence.Field4ChoiceType choice = TestSequence.Field4ChoiceType ();
choice.selectField2(0.5);
sequence.Field4 = choice;
IEncoder encoder = CoderFactory.getInstance().newEncoder("BER");
ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
encoder.encode< TestSequence >(sequence, outputStream);

...
// Decoding for Java
IDecoder decoder = CoderFactory.getInstance().newDecoder("BER");
TestSequence seq = decoder.decode< TestSequence>(stream);
System.out.println(seq.Field1);
if(seq.isField3Present()) {
        System.out.println(seq.Field3);
}
...
```

6. BinaryNotes Message Queues

BinaryNotes Message Queues (BNMQ) v1.0 is the main and new feature in BinaryNotes v1.3.
Functionality:

- Compact ASN.1 encoded messages (default encoding is PER/Unaligned)
- Flexible subscription procedure with filtering on server support
- Supplier/Consumer can create dynamic as on server and client.
- Persistence (mandatory) messages and subscription supported (based on HSQLDB for Java, and SQLite for .net)
- Framework hide all networking infrastructure. For BNMQ version 1.0 only supported auto-restore connection feature. But is planned for more progressive Fault-Tolerant and Load-Balancing Feature (v1.1).

Note: Section has been updated after released 1.3.

e-mail : abdulla.abdurakhmanov@gmail.com

7. License

The BinaryNotes Library and Message Queues is made available subject to the terms of GNU Lesser General Public License Version 2. Please read original license from http://www.gnu.org/copyleft/lgpl.html or from distribution package (Dist\licenses).

The BinaryNotes Compiler is made available subject to the terms of GNU General Public License Version 2. Please read original license from http://www.gnu.org/copyleft/gpl.html or from distribution package (Dist\licenses).

Third Party Code. Additional copyright notices and license terms applicable to portions of the Software are set forth in the Dist\licenses\3rdyparty\ directory.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

e-mail : abdulla.abdurakhmanov@gmail.com