Value Conversions

JiBX defines a default set of formats used for converting property values to and from text strings. These format definitio are effectively defined in a context outside the root **binding** element of the binding definition. Most of the conversions a based on the W3C XML Schema datatype definitions, but do differ from the standard schema definitions in that leading a trailing whitespace characters are not stripped from values (meaning the default conversions will report errors for valid schema values in cases where leading or trailing whitespace is used). If this causes problems for your data, the best solution for now is to define custom conversions which strip the leading and trailing whitespace from values. The default formats for simple values are shown in the following table (also see the Date Time Conversions page, for the standard conversions relating to date/time values).

Default Formats

Type	Format Label	Conversion
`byte`	byte.default	Converts primitive `byte` values to and from the schema byte representatio (integer values in the range of -128 to +127).
`char`	char.default	Converts primitive `char` values to and from the schema unsigned short representation (integer values in the range of 0 to 65535). Although this is the default for `char` primitives, JiBX also supports an alternate conversion a single-character text values using the named format **char.string** (allowing the conversion to be applied on a specific value) and a pair of conversion methods. To make the single-character text conversion the default in your binding, just specify the format as a child of the root **binding** element as follows: `<format type="char" serializer="org.jibx.runtime.Utility.serializeCharString" deserializer="org.jibx.runtime.Utility.deserializeCharString`
`double`	double.default	Converts primitive `double` values to and from the schema double representation (IEEE double-precision floating point).
`float`	float.default	Converts primitive `float` values to and from the schema float representati (IEEE single-precision floating point).
`int`	int.default	Converts primitive `int` values to and from the schema int representation (integer values in normal 32-bit signed value range).
`long`	long.default	Converts primitive `long` values to and from the schema long representatio (integer values in normal 64-bit signed value range).
`short`	short.default	Converts primitive `short` values to and from the schema short representation (integer values in normal 16-bit signed value range).
`boolean`	boolean.default	Converts primitive `boolean` values to and from the schema boolean representation ("true" or "false", or equivalently "1" or "0" - the former values are always used when converting to text).
`byte[]`	byte-array.default	Converts byte arrays to and from the schema base64 representation to allo arbitrary binary data.
`java.lang.String`	String.default	Identity converter for `java.lang.String` instances.
`org.jibx.runtime.QName`	QName.default	Converter for namespace-qualified names using `org.jibx.runtime.QName`
`java.lang.Boolean`	Boolean.default	Converter for `java.lang.Boolean` instances to and from the schema boolean representation. As with **boolean.default**, this always uses the values "true" and "false" when converting to text.
`java.lang.Object`	Object.default	

		Converts any object to a string representation by using the `toString()` method, and converts a string to any object by using a constructor that tak a single argument of type `java.lang.String`. If an optional value is not present when unmarshalling a `null` value is stored to the object reference. This is the default format conversion used for any object type without a mo specific conversion defined.

These are the conversions you'll get unless you specify otherwise. If you use a **format** element to override one of these defaults you'll still be able to access the default by using the defined label for that format. Defining your own custom serialization formats is easy, basically requiring only a pair of static methods to convert to and from `String` representations. See the format element description for details of defining your own conversions. The `org.jibx.runtime.Utility` class defines a variety of conversion methods, including those used to implement the standard conversions listed in the above table, so it's a good starting point for developing your own variations.

`java.math.BigDecimal` is a special case. For 1.3 and 1.4.1 JDKs the default `java.lang.Object` conversion from the above table worked fine. In the 1.5 JDK Sun made a non-backward-compatible change to the format of the `BigDecimal.toString()` output, which meant that the resulting text does not always match the decimal schema representation (due to the use of exponents). Because of this JDK compatibility failure JiBX has an added optional conversion for `BigDecimal` which uses a method added in 1.5 JDK to obtain decimal-compatible output:

Non-default Formats

Type	Format Label	Conversion
`java.math.BigDecimal`	BigDecimal.java5	Converts `BigDecimal` values to and from the schema decimal representatic (string of digits with optional leading sign and optional embedded decimal point). This conversion can **only** be used for 1.5 and later JDKs.

In addition to the conversions listed in the tables, JiBX 1.1 and later support automatic conversion of Java 5 enum value to and from strings.