

# APS JSON Library

## User Guide

1.0.0

Tommy Svensson

Copyright © 2012 Natusoft AB

APSJSONLib .....	1
<i>Changes</i> .....	<i>1</i>
0.10.0 .....	1
<i>APIs</i> .....	<i>1</i>

# APJSONLib

This is a library (exports all its packages and provides no service) for reading and writing JSON. It can also write a JavaBean object as JSON and take a JSON value or inputstream containing JSON and produce a JavaBean.

This basically provides a class representing each JSON type: JSONObject, JSONString, JSONNumber, JSONBoolean, JSONArray, JSONNull, and a JSONValue class that is the common base class for all the other. Each class knows how to read and write the JSON type it represents. Then there is a JavaToJSON and a JSONToJava class with static methods for converting back and forth. This mapping is very primitive. There has to be one to one between the JSON and the Java objects.

## Changes

---

### 0.10.0

`readJSON(...)` in the **JSONValue** base class now throws `JSONEOFException` (extends `IOException`) on EOF. The reason for this is that internally it reads characters which cannot return -1 or any non JSON data valid char to represent EOF. Yes, it would be possible to replace `char` with `Character`, but that will have a greater effect on existing code using this lib. If an `JSONEOFException` comes and is not handled it is still very much more clear what happened than a `NullPointerException` would be!

## APIs

---

Complete javadocs can be found at <http://apidoc.natusoft.se/APJSONLib/>.

```
public class JSON [se.natusoft.osgi.aps.json] {
```

This is the official API for reading and writing JSON values.

```
public static void read(InputStream jsonIn, APSHandler<APResult<JSONValue>>  
resultHandler)
```

Reads any JSON object from the specified *InputStream*.

*Parameters*

*jsonIn*- The *InputStream* to read from.

*resultHandler*- The handler to call with result.

```
public static JSONValue read(InputStream jsonIn, JSONErrorHandler errorHandler)
```

Reads any JSON object from the specified *InputStream*.

*Returns*

A *JSONValue* subclass. Which depends on what was found on the stream.

*Parameters*

*jsonIn*- The *InputStream* to read from.

*errorHandler*- An implementation of this interface should be supplied by the user to handle any errors during JSON parsing.

*Throws*

*APIOException*- on any IO failures.

```
public static void write(OutputStream jsonOut, JSONValue value) throws APIOException
```

Writes a *JSONValue* to an *OutputStream*. This will write compact output by default.

*Parameters*

*jsonOut*- The *OutputStream* to write to.

*value*- The value to write.

*Throws*

*APIOException*- on failure.

**public static void write(OutputStream jsonOut, JSONValue value, boolean compact, APSHandler<APSResult<Void>> resultHandler)**

Writes a *JSONValue* to an *OutputStream*. This will write compact output by default.

*Parameters*

*jsonOut*- The *OutputStream* to write to.

*value*- The value to write.

*resultHandler*- handler for result. only *success()* or *failure()* is relevant.

**public static void write(OutputStream jsonOut, JSONValue value, boolean compact) throws APIOException**

Writes a *JSONValue* to an *OutputStream*.

*Parameters*

*jsonOut*- The *OutputStream* to write to.

*value*- The value to write.

*compact*- If true the written JSON is made very compact and hard to read but produce less data.

*Throws*

*APIOException*- on IO problems.

}

---

**public JSONArrayProvider()**

Creates a new *JSONArray* for writing JSON output.

**public JSONArrayProvider(JSONErrorHandler errorHandler)**

Creates a new *JSONArray* for reading JSON input and writing JSON output.

*Parameters*

*errorHandler*- The error handler to use.

**public void addValue(JSONValueProvider value)**

Adds a value to the array.

*Parameters*

*value*- The value to add.

**public List<se.natusoft.osgi.aps.api.misc.json.model.JSONValue> getAsList()**

Returns the array values as a List.

**public <T extends se.natusoft.osgi.aps.api.misc.json.model.JSONValue> List<T>  
getAsList(Class<T> type)**

Returns the array values as a list of a specific type.

*Returns*

*A list of specified type if type is the same as in the list.*

*Parameters*

*type*- The class of the type to return values as a list of.

*<T>*- One of the JSONValue subclasses.

}

---

**public JSONBooleanProvider(boolean value)**

Creates a new JSONBoolean instance for writing JSON output.

*Parameters*

*value*- The value for this boolean.

**public JSONBooleanProvider(JSONErrorHandler errorHandler)**

Creates a new JSONBoolean instance for reading JSON input or writing JSON output.

*Parameters*

*errorHandler*- The error handler to use.

**public void setBooleanValue(boolean value)**

Sets the value of this boolean.

*Parameters*

*value*- The value to set.

**public boolean getAsBoolean()**

Returns the value of this boolean.

**public String toString()**

Returns the value of this boolean as a String.

}

---

**public JSONNullProvider()**

Creates a new JSONNull instance for writing JSON output.

**public JSONNullProvider(JSONErrorHandler errorHandler)**

Creates a new JSONNull instance for reading JSON input or writing JSON output.

*Parameters*

*errorHandler- The error handler to use.*

**public String toString()**

*Returns*

*as String.*

}

---

**public JSONNumberProvider(Number value)**

Creates a new JSONNumber instance for writing JSON output.

*Parameters*

*value- The numeric value.*

**public JSONNumberProvider(JSONErrorHandler errorHandler)**

Creates a new JSONNumber instance for reading JSON input or writing JSON output.

*Parameters*

*errorHandler- The error handle to use.*

**public Number toNumber()**

Returns the number as a Number.

**public float toFloat()**

Returns the number as a float value.

**public int toInt()**

Returns the number as an int value.

**public long toLong()**

Returns the number as a long value.

**public short toShort()**

Returns the number as a short value.

**public byte toByte()**

Returns the number as a byte value.

**public String toString()**

*Returns*

*number as String.*

**public Object to(Class type)**

Returns the number as a value of the type specified by the type parameter.

*Parameters*

*type-* The type of the returned number.

}

---

**public JSONObjectProvider()**

Creates a JSONObject instance for writing JSON output.

**public JSONObjectProvider(JSONErrorHandler errorHandler)**

Creates a new JSONObject instance for reading JSON input or writing JSON output.

*Parameters*

*errorHandler-* The error handler to use.

**public void setValue(se.natusoft.osgi.aps.api.misc.json.model.JSONString name,**

### **JSONValueProvider value)**

Adds a property to this JSONObject instance.

#### *Parameters*

*name-* The name of the property.

*value-* The property value.

### **public void setValue(String name, se.natusoft.osgi.aps.api.misc.json.model.JSONValue value)**

Adds a property to this JSONObject instance.

#### *Parameters*

*name-* The name of the property.

*value-* The property value.

}

---

### **public JSONStringProvider(String value)**

Creates a new JSONString for writing JSON output.

#### *Parameters*

*value-* The value of this JSONString.

### **public JSONStringProvider(JSONErrorHandler errorHandler)**

Creates a new JSONString for reading JSON input and writing JSON output.

#### *Parameters*

*errorHandler-* The error handler to use.

}

---

### **protected JSONValueProvider()**

Creates a new JSONValue.

### **protected JSONValueProvider(JSONErrorHandler errorHandler)**



Creates a new JSONValue

**protected abstract void readJSON(char c, JSONReader reader) throws APSIOException**

This will read the value from an input stream.

*Parameters*

*c- The first character already read from the input stream.*

*reader- The reader to read from.*

*Throws*

*APSIOException- on IO failure.*

**protected abstract void writeJSON(JSONWriter writer, boolean compact) throws APSIOException**

This will write the data held by this JSON value in JSON format on the specified stream.

*Parameters*

*writer- A JSONWriter instance to write with.*

*compact- If true write the JSON as compact as possible. false means readable, indented.*

*Throws*

*APSIOException- On IO failure.*

**protected JSONErrorHandler getErrorHandler()**

*Returns*

*The user supplied error handler.*

***/\*package\*/***

Reads and resolves what JSON type is the next in the input and returns it.

*Returns*

*The read JSONValue.*

*Parameters*

*c- The first already read character.*

*reader- The reader to read from.*

*errorHandler- The user supplied error handler.*

*Throws*

*APSIOException- on IOFailure.*

**protected void fail(String message, Throwable cause)**

Fails the job.

#### *Parameters*

*message*- The failure message.

*cause*- An eventual cause of the failure. Can be null.

#### **protected void fail(String message)**

Fails the job.

#### *Parameters*

*message*- The failure message.

#### **public void readJSON(InputStream is) throws APSIOException**

This will read the value from an input stream.

#### *Parameters*

*is*- The input stream to read from.

#### *Throws*

*APSIOException*- on IO failure.

#### **public void writeJSON(OutputStream os) throws APSIOException**

This writes JSON to the specified OutputStream.

#### *Parameters*

*os*- The outoutStream to write to.

#### *Throws*

*APSIOException*- on IO failure.

#### **public void writeJSON(OutputStream os, boolean compact) throws APSIOException**

This writes JSON to the specified OutputStream.

#### *Parameters*

*os*- The outoutStream to write to.

*compact*- If true write JSON as compact as possible. If false write it readable with indents.

#### *Throws*

*APSIOException*- on IO failure.

#### ***/\*package\*/***

Method for creating a JSONString instance.

#### *Parameters*

*errorHandler*- The user error handler.

#### ***/\*package\*/***

Method for creating a JSONNumber instance.

*Parameters*

*errorHandler- The user error handler.*

***/\*package\*/***

Method for creating a JSONNull instance.

*Parameters*

*errorHandler- The user error handler.*

***/\*package\*/***

Method for creating a JSONBoolean instance.

*Parameters*

*errorHandler- The user error handler.*

***/\*package\*/***

Method for creating a JSONArray instance.

*Parameters*

*errorHandler- The user error handler.*

***/\*package\*/***

Method for creating a JSONObject instance.

*Parameters*

*errorHandler- The user error handler.*

**protected JSONReader(PushbackReader reader, JSONErrorHandler errorHandler)**

Creates a new JSONReader instance.

*Parameters*

*reader- The PushbackReader to read from.*

*errorHandler- The handler for errors.*

**protected char getChar() throws APSIOException**

Returns the next character on the specified input stream, setting EOF state checkable with isEOF().

*Throws*

*APSIOException- on IO problems.*

For subclasses to use in writeJSON(JSONWriter writer).

**protected JSONWriter(Writer writer)**

Creates a new JSONWriter instance.

*Parameters*

*writer*- The writer to write to.

**protected void write(String json) throws APSIOException**

Writes JSON output.

*Parameters*

*json*- The JSON output to write.

*Throws*

*APSIOException*- on IO failure.

}

---

**public BeanInstance(Object modellInstance)**

Creates a new ModellInstance.

*Parameters*

*modellInstance*- The model instance to wrap.

**public Object getModellInstance()**

Returns the test model instance held by this object.

**public List<String> getSettableProperties()**

Returns a list of settable properties.

**public List<String> getGettableProperties()**

Returns a list of gettable properties.

**public void setProperty(String property, Object value) throws JSONConversionException**

Sets a property

*Parameters*

*property*- The name of the property to set.

*value*- The value to set with.

*Throws*

*JSONConversionException*- on any failure to set the property.

**public Object getProperty(String property) throws JSONException**

Returns the value of the specified property.

*Returns*

*The property value.*

*Parameters*

*property-* The property to return value of.

*Throws*

*JSONException-* on failure (probably bad property name!).

**public Class getPropertyType(String property) throws JSONException**

Returns the type of the specified property.

*Returns*

*The class representing the property type.*

*Parameters*

*property-* The property to get the type for.

*Throws*

*JSONException-* if property does not exist.

}

---

public class **CollectingErrorHandler** implements JSONErrorHandler  
[se.natusoft.osgi.aps.json.tools] {

Utility implementation of JSONErrorHandler.

**public CollectingErrorHandler(boolean printWarnings)**

*Parameters*

*printWarnings-* If true warnings will be printed to stderr.

**public boolean hasMessages()**

*Returns*

*true if there are any messages.*

**public String toString()**

*Returns*

*All messages as one string.*

}

---

**public static JSONObjectProvider convertObject(Object javaBean) throws JSONConversionException**

Converts a JavaBean object into a *JSONObject*.

*Returns*

*A JSONObject containing all values from the JavaBean.*

*Parameters*

*javaBean*- The JavaBean object to convert.

*Throws*

*JSONConversionException*- on converting failure.

**public static JSONObjectProvider convertObject(JSONObjectProvider jsonObject, Object javaBean) throws JSONConversionException**

Converts a JavaBean object into a *JSONObject*.

*Returns*

*A JSONObject containing all values from the JavaBean.*

*Parameters*

*jsonObject*- The jsonObject to convert the bean into or null for a new JSONObject.

*javaBean*- The JavaBean object to convert.

*Throws*

*JSONConversionException*- on converting failure.

**public static JSONValueProvider convertValue(Object value)**

Converts a value from a java value to a *JSONValue*.

*Returns*

*The converted JSONValue.*

*Parameters*

*value*- The java value to convert. It can be one of String, Number, Boolean, null, JavaBean, or an array of those.

}

---

**public class JSONConversionException** extends RuntimeException  
[se.natusoft.osgi.aps.json.tools] {

This exception is thrown on failure to convert from JSON to Java or Java to JSON.

Almost all exceptions within the APS services and libraries extend either *APSEException* or *APSRuntimeException*. I decided to just extend *RuntimeException* here to avoid any other dependencies for this library since it can be useful outside of APS and can be used as any jar if not deployed in OSGi container.

**public JSONConversionException(final String message)**

Creates a new *JSONConversionException*.

*Parameters*

*message*- The exception message

**public JSONConversionException(final String message, final Throwable cause)**

Creates a new *JSONConversionException*.

*Parameters*

*message*- The exception message

*cause*- The cause of this exception.

}

---

**public class JSONMapConv** [se.natusoft.osgi.aps.json.tools] {

This converts between a Java Map and JSON. Do note that this of course uses this library to read and write JSON, but this specific public API only deals with Java and JSON as String or on/in a stream. [p/](#) This class becomes more useful when used from Groovy since the latter provides much nicer usage of data in Maps. Yes, I know about JSONSlurper and JSONBuilder in Groovy. Those however does not work with `@CompileStatic`. Maps does.

**public static Map<String, Object> jsonObjectToMap(String json) throws APSIOException**

This takes a String containing a JSON object and returns it as a Map.

*Parameters*

*json*- The JSON content to convert to a Map.

*Throws*

*APSIOException*- on failure.

**public static Map<String, Object>  
jsonObjectToMap(se.natusoft.osgi.aps.api.misc.json.model.JSONObject jsonObject)**

This takes a JSONObject and returns a Map.

*Returns*

*The converted Map.*

*Parameters*

*jsonObject*- The JSONObject to convert to a Map.

**public static String mapToJSONObjectString(Map<String, Object> map) throws APSIOException**

This takes a Map (as created by jsonObjectToMap(...)) and returns a JSON String.

*Parameters*

*map*- The Map to convert to JSON.

*Throws*

*APSIOException*- on I/O failures.

**public static se.natusoft.osgi.aps.api.misc.json.model.JSONObject  
mapToJSONObject(Map<String, Object> map)**

Converts a `Map<String, Object>` to a `JSONObject`.

*Returns*

*A converted JSONObject.*

*Parameters*

*map*- The Map to convert.

}

---

**public class JSOToJava** [se.natusoft.osgi.aps.json.tools] {

Creates a JavaBean instance and copies data from a JSON value to it.

The following mappings are made in addition to the expected ones:

- *JSONArray* only maps to an array property.
- Date properties in bean are mapped from *JSONString* "yyyy-MM-dd HH:mm:ss".
- Enum properties in bean are mapped from *JSONString* which have to contain enum constant name.

**public static <T> T convert(InputStream jsonStream, Class<T> javaClass) throws  
APSIOException, JSONConversionException**

Returns an instance of a java class populated with data from a json object value read from a stream.

*Returns*

*A populated instance of javaClass.*

*Parameters*

*jsonStream*- The stream to read from.

*javaClass*- The java class to instantiate and populate.

*Throws*



*APIOException- on IO failures.*

*JSONConversionException- On JSON to Java failures.*

**public static <T> T convert(String json, Class<T> javaClass) throws APIOException, JSONConversionException**

Returns an instance of a java class populated with data from a json object value read from a String containing JSON.

*Returns*

*A populated instance of javaClass.*

*Parameters*

*json- The String to read from.*

*javaClass- The java class to instantiate and populate.*

*Throws*

*APIOException- on IO failures.*

*JSONConversionException- On JSON to Java failures.*

**public static <T> T convert(JSONValue json, Class<T> javaClass) throws JSONConversionException**

Returns an instance of java class populated with data from json.

*Returns*

*A converted Java object.*

*Parameters*

*json- The json to convert to java.*

*javaClass- The class of the java object to convert to.*

*Throws*

*JSONConversionException- On failure to convert.*

}

---

**public class SystemOutErrorHandler** implements JSONErrorHandler  
[se.natusoft.osgi.aps.json.tools] {

A simple implementation of *JSONErrorHandler* that simply displays messages on System.out and throws a *RuntimeException* on fail. This is used by the tests. In a non test case another implementation is probably preferred.

}

---