

# APS JSON Library

User Guide

Version: 0.9.0

Author: Tommy Svensson

Copyright © 2013 Natusoft AB

## Table of Contents

<b>1 APSJSONLib</b>	<b>1</b>
1.1 APIs	1

# 1 APSJSONLib

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.

This does not try to be an alternative to Jackson! This is used internally by other services.

## 1.1 APIs

---

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

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

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

*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

*IOException* - on any IO failures.

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

*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

*IOException* - on failure.

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

*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

*IOException*

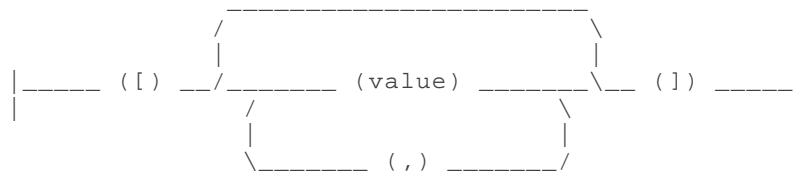
}

---

**public class JSONArray** extends JSONValue [se.natusoft.osgi.aps.json] {

*This class is based on the structure defined on <http://www.json.org/>.*

*This represents the "array" diagram on the above mentioned web page:*



@author Tommy Svensson

**public JSONArray()**

*Creates a new JSONArray for writing JSON output.*

**public JSONArray(JSONErrorHandler errorHandler)**

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

#### Parameters

*errorHandler*

**public void addValue(JSONValue value)**

*Adds a value to the array.*

#### Parameters

*value* - The value to add.

**public List<JSONValue> getAsList()**

*Returns the array values as a List.*

**public <T extends 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 class JSONBoolean** extends JSONValue [se.natusoft.osgi.aps.json] {

*This class is based on the structure defined on <http://www.json.org/>. @author Tommy Svensson*

**public JSONBoolean(boolean value)**

*Creates a new JSONBoolean instance for writing JSON output.*

#### Parameters

*value* - The value for this boolean.

**public JSONBoolean(JSONErrorHandler errorHandler)**

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

#### Parameters

*errorHandler*

**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 interface JSONErrorHandler** [se.natusoft.osgi.aps.json] {

*This is called on warnings or failures. @author Tommy Svensson*

#### **void warning(String message)**

*Warns about something.*

#### Parameters

*message* - The warning message.

#### **void fail(String message, Throwable cause) throws RuntimeException**

*Indicate failure.*

#### Parameters

*message* - The failure message.

*cause* - The cause of the failure. Can be null!

#### Throws

*RuntimeException* - This method must throw a RuntimeException.

}

---

**public class JSONNull** extends JSONValue [se.natusoft.osgi.aps.json] {

*This class is based on the structure defined on <http://www.json.org/>. @author Tommy Svensson*

**public JSONNull()**

*Creates a new JSONNull instance for writing JSON output.*

**public JSONNull(JSONErrorHandler errorHandler)**

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

**Parameters**

*errorHandler*

**public String toString()****Returns**

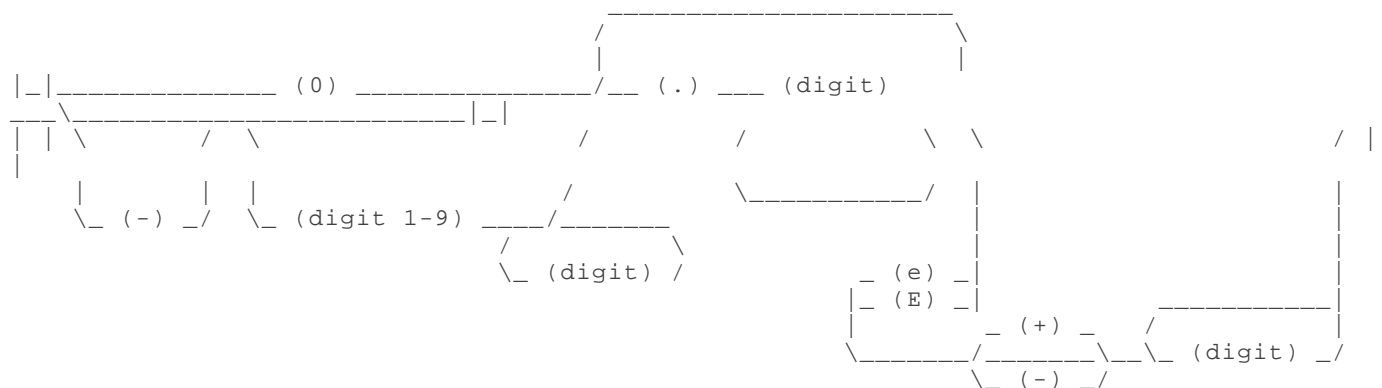
*as String.*

}

**public class JSONNumber** extends JSONValue [se.natusoft.osgi.aps.json] {

*This class is based on the structure defined on <http://www.json.org/>.*

*This represents the "number" diagram on the above mentioned web page:*



@author Tommy Svesson

**public JSONNumber(Number value)**

*Creates a new JSONNumber instance for writing JSON output.*

### Parameters

*value* - The numeric value.

### **public JSONNumber(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 double toDouble()**

*Returns the number as a double value.*

### **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.

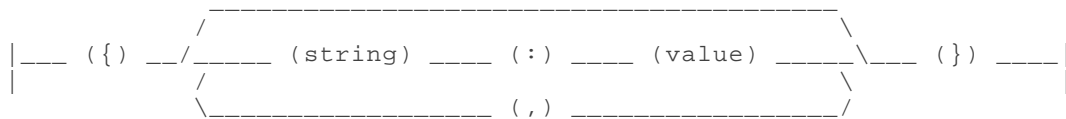
}

---

```
public class JSONObject extends JSONValue [se.natusoft.osgi.aps.json] {
```

*This class is based on the structure defined on <http://www.json.org/>.*

*It represents the "object" diagram on the above mentioned web page:*



*This is also the starting point.*

*To write JSON, create a new JSONObject (new JSONObject()) and call addProperty(name, value) for children. Then do jsonObj.writeJSON(outputStream).*

*To read JSON, create a new JSONObject (new JSONObject(jsonErrorHandler)) and then do jsonObj.readJSON(inputStream). Then use getProperty(name) to extract children. @author Tommy Svensson*

### **public JSONObject()**

*Creates a JSONObject instance for writing JSON output.*

### **public JSONObject(JSONErrorHandler errorHandler)**

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

### Parameters

*errorHandler*

### **public Set<JSONString> getPropertyNames()**

*Returns the names of the available properties.*

### **public JSONValue getProperty(JSONString name)**

*Returns the named property.*



```
|__ (f) (formfeed) _____|
|__ (n) (newline) _____|
|__ (r) (carriage return) _____|
|__ (t) (orizontal tab) _____|
\__ (u) (4 hexadecimal digits) _____/
```

@author Tommy Svensson

## public JSONString(String value)

*Creates a new JSONString for writing JSON output.*

### Parameters

*value* - The value of this JSONString.

## public JSONString(JSONErrorHandler errorHandler)

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

### Parameters

*errorHandler*

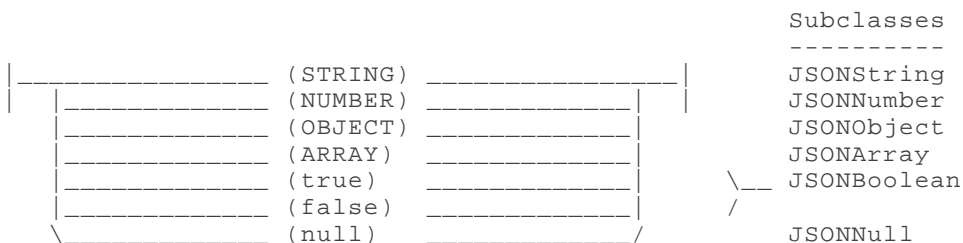
```
}
```

---

## public abstract class JSONValue [se.natusoft.osgi.aps.json] {

*This class is based on the structure defined on <http://www.json.org/>.*

*This is a base class for all other JSON\* classes. It represents the "value" diagram on the above mentioned web page:*



@author Tommy Svensson

**protected JSONValue()**

*Creates a new JSONValue.*

**protected JSONValue(JSONErrorHandler errorHandler)**

*Creates a new JSONValue*

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

*This will read the vale from an input stream.*

**Returns**

*the last character read.*

**Parameters**

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

*reader* - The reader to read from.

**Throws**

*IOException* - on IO failure.

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

*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**

*IOException* - On IO failure.

**protected JSONErrorHandler getErrorHandler()****Returns**

*The user supplied error handler.*

**protected void warn(String message)**

*Provide a warning.*

#### Parameters

*message* - The warning message.

#### **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 IOException**

*This will read the value from an input stream.*

#### Parameters

*is* - The input stream to read from.

#### Throws

*IOException* - on IO failure.

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

*This writes JSON to the specified OutputStream.*

#### Parameters

*os* - The outputStream to write to.

#### Throws

*IOException* - on IO failure.

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

*This writes JSON to the specified OutputStream.*

**Parameters**

*os* - The *OutputStream* to write to.

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

**Throws**

*IOException* - on IO failure.

**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 IOException**

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

**Throws**

*IOException* - on IO problems.

**protected char getChar(boolean handleEscapes) throws IOException**

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

**Parameters**

*handleEscapes* - If true then \\* escape character are handled.

**Throws**

*IOException* - on IO problems.

**protected void ungetChar(char c) throws IOException**

*Unreads the specified character so that the next call to getNextChar() will return it again.*

*Parameters*

*c* - The character to unget.

**protected char skipWhitespace(char c) throws IOException**

*Skips whitespace returning the first non whitespace character. This also sets the EOF flag.*

*Parameters*

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

*Throws*

*IOException*

**protected char skipWhitespace() throws IOException**

*Skips whitespace returning the first non whitespace character. This also sets the EOF flag.*

*Throws*

*IOException*

**protected char readUntil(String until, char c, StringBuilder sb, boolean handleEscapes) throws IOException**

*Reads until any of a specified set of characters occur.*

*Returns**Parameters*

*until* - The characters to stop reading at. The stopping character will be returned unless EOF.

*c* - The first preread character.

*sb* - If not null read characters are added to this. The stopping character will not be included.

*handleEscapes* - True if we are reading a string that should handle escape characters.

*Throws*

*IOException*

**protected char readUntil(String until, StringBuilder sb, boolean string) throws IOException**

*Reads until any of a specified set of characters occur.*

### Parameters

*until* - The characters to stop reading at. The stopping character will be returned unless EOF.

*sb* - If not null read characters are added to this. The stopping character will not be included.

*string* - True if we are reading a string that should be escaped.

### Throws

*IOException*

### **protected char readUntil(String until, StringBuilder sb) throws IOException**

*Reads until any of a specified set of characters occur.*

### Parameters

*until* - The characters to stop reading at. The stopping character will be returned unless EOF.

*sb* - If not null read characters are added to this. The stopping character will not be included.

### Throws

*IOException*

### **protected boolean checkValidChar(char c, String validChars)**

*Returns true if c is one of the characters in validChars.*

### Parameters

*c* - The character to check.

*validChars* - The valid characters.

### **protected void assertChar(char a, char e, String message)**

*Asserts that char a equals expected char c.*

### Parameters

*a* - The char to assert.

*e* - The expected value.

*message* - Failure message.

### **protected void assertChar(char a, String expected, String message)**



*Asserts that char a equals expected char c.*

#### Parameters

*a* - The char to assert.

*expected* - String of valid characters.

*message* - Failure message.

protected static class **JSONWriter** [se.natusoft.osgi.aps.json] {

*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 IOException

*Writes JSON output.*

#### Parameters

*json* - The JSON output to write.

#### Throws

*IOException* - on IO failure.

#### protected void writeLn(String json) throws IOException

*Writes JSON output plus a newline.*

#### Parameters

*json* - The JSON output to write.

#### Throws

*IOException*

}

---

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

*This wraps a Java Bean instance allowing it to be populated with data using `setProperty(String, Object)` methods handling all reflection calls.*

### **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 JSONConversionException**

*Returns the value of the specified property.*

#### *Returns*

*The property value.*

#### *Parameters*

*property* - The property to return value of.

#### *Throws*

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

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

*Returns the type of the specified property.*

*Returns*

*The class representing the property type.*

*Parameters*

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

*Throws*

*JSONConversionException - if property does not exist.*

}

---

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

*Takes a JavaBean and produces a JSONObject.*

**public static JSONObject 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 JSONObject convertObject(JSONObject 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 JSONValue 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 JSONTToJava [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: <ul <li JSONArray only maps to an array property. li <li Date properties in bean are mapped from `JSONString` "yyyy-MM-dd HH:mm:ss" li <li Enum properties in bean are mapped from `JSONString` which have to contain enum constant name. li ul*

```
public static <T> T convert(InputStream jsonStream, Class<T> javaClass) throws IOException,
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*

*`IOException` - on IO failures.*

*`JSONConversionException` - On JSON to Java failures.*

```
public static <T> T convert(String json, Class<T> javaClass) throws IOException, 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*

*IOException* - 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.*

}

---