Network Working Group   B. Muschett
Internet-Draft   R. Salz
Expires: November 3, 2011   M. Schenker
                                                                     IBM
                                                             May 2, 2011

JSONx, an XML Encoding for JSON
draft-rsalz-jsonx-00.txt

Abstract

This document specifies a mapping between JSON (RFC 4627) and XML. The mapping maintains a high degree of fidelity. It is used by several IBM products.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies a mapping between JSON [RFC4627] and XML, known as JSONx. The mapping maintains a high degree of fidelity. It is used by several IBM products.

JSONx is specified using the terms from the XML Infoset [REC-xml-infoset], serialized as XML 1.0 [REC-xml]. The Infoset terms "Element Information Item," "Attribute Information Item," and "Character Information Item," are shortened to "element," "attribute," and "characters" respectively. For example, when this specification uses the term "element," it is referring to an Element Information Item, and when it uses the term "attribute," it is referring to an Attribute Information Item.

2. Conversion Rules

JSON identifiers are represented by the string contents of the "name" attribute. Most Unicode characters other than backspace (Unicode code point U+0008) and form feed (U+000C) are valid within identifiers, as long as they are properly escaped (for example, \unnnn). When JSONx is serialized as XML documents, character and/or entity references may need to be used for special characters. Examples of this include ampersand (U+0026), less-then sign (U+003C), and any characters not representable in the document’s encoding.

Use of backspace, formfeed, or NUL (U+0000) is undefined.

2.1. root element

The root element is either a <json:object> or a <json:array> element with the following namespace declaration:

Prefix	Namespace URI
json	http://www.ibm.com/xmlns/prod/2009/jsonx

All elements defined in this document are in that namespace.

2.2. object

A JSON object becomes a <json:object> element. If the object denotes a property within a JSON object, JSONx encodes a name attribute whose value is assigned the property name. The child elements depend on the properties of the JSON object. To improve readability, whitespace characters may be added between child elements. Object elements are ordered according to their document order.

```
{ "Ticker" : "IBM" }

<json:object>
    <json:string name="Ticker">IBM</json:string>
</json:object>
```

2.3. array

A JSON array becomes a <json:array> element. If the array denotes a property within a JSON object, JSONx encodes a name attribute whose value is assigned the property name. The child elements depend on the values in the array. To improve readability, whitespace characters may be added between child elements.

```
"phoneNumbers": [
    "212 555-1111",
    "212 555-2222"
]
```

```
<json:array name="phoneNumbers">
    <json:string>212 555-1111</json:string>
    <json:string>212 555-2222</json:string>
</json:array>
```

2.4. boolean

A JSON boolean becomes a <json:boolean> element. If the boolean denotes a property within a JSON object, JSONx encodes a name attribute whose value is assigned the property name. The boolean value is character data as either true or false.

```
"remote": false
```

```
<json:boolean name="remote">false</json:boolean>
```

2.5. string

A JSON string becomes a <json:string> element. If the string denotes a property within a JSON object, JSONx encodes a name attribute whose value is assigned the property name. The string value is character data.

Use of the Unicode code points U+0000, U+0008, and U+000C is undefined.

```
"name": "John Smith"
```

```
<json:string name="name">John Smith</json:string>
```

2.6. number

A JSON number becomes a <json:number> element. If the number denotes a property within a JSON object, JSONx encodes a name attribute whose value is assigned the property name. The number value is character data.

```
"height": 62.4
```

```
<json:number name="height">62.4</json:number>
```

2.7. null

JSON value of null becomes a <json:null> element. If the null value denotes a property within a JSON object, JSONx encodes a name attribute whose value is assigned the property name. This element has no content.

```
"additionalInfo": null
```

```
<json:null name="additionalInfo" />
```

3. Extended Example

The following example document is a sample of the JSON structure.

```
{
    "name": "John Smith"
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021,
    },
    "phoneNumbers": [
        "212 555-1111",
        "212 555-2222"
    ],
    "additionalInfo": null,
    "remote": false,
    "height": 62.4,
    "ficoScore": "> 640"
}
```

The following output is the result of the transformed document as JSONx.

```
<?xml version="1.0" encoding="UTF-8"?>
<json:object xmlns:json="http://www.ibm.com/xmlns/prod/2009/jsonx">
    <json:string name="name">John Smith</json:string>
    <json:object name="address">
        <json:string name="streetAddress">21 2nd Street</json:string>
        <json:string name="city">New York</json:string>
        <json:string name="state">NY</json:string>
        <json:number name="postalCode">10021</json:number>
    </json:object>
    <json:array name="phoneNumbers">
        <json:string>212 555-1111</json:string>
        <json:string>212 555-2222</json:string>
    </json:array>
    <json:null name="additionalInfo" />
    <json:boolean name="remote">false</json:boolean>
    <json:number name="height">62.4</json:number>
    <json:string name="ficoScore">&gt; 640</json:string>
</json:object>
```

4. Normative References

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[REC-xml] Yergeau, F., Paoli, J., Bray, T., Sperberg-McQueen, C., and E. Maler, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium Recommendation REC-xml-20060816, August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>.

[REC-xml-infoset] Cowan, J. and R. Tobin, "XML Information Set (Second Edition)", World Wide Web Consortium Recommendation REC-xml-infoset-20040204, February 2004, <http://www.w3.org/TR/2004/REC-xml-infoset-20040204>.

Appendix A. Schema (not normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ibm.com/xmlns/prod/2009/jsonx"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="http://www.ibm.com/xmlns/prod/2009/jsonx">

    <xsd:simpleType name="jsonnumbertype">
        <xsd:restriction base="xsd:token">
            <xsd:pattern value="[-]?(0|[1-9][0-9]*)(\.[0-9]+)?([eE][-+]?[0-9]+)?" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:element name="object" type="tns:anyElement" />
    <xsd:element name="array" type="tns:anyElement" />
    <xsd:element name="string" type="tns:stringElement" />
    <xsd:element name="number" type="tns:numberElement" />
    <xsd:element name="boolean" type="tns:booleanElement" />
    <xsd:element name="null" type="tns:emptyElement" />

    <xsd:complexType name="anyElement">
        <xsd:sequence>
            <xsd:any minOccurs="0" maxOccurs="unbounded"
                    namespace="##targetNamespace" processContents="strict" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" />
    </xsd:complexType>

    <xsd:complexType name="emptyElement">
        <xsd:attribute name="name" type="xsd:string" />
    </xsd:complexType>

    <xsd:complexType name="stringElement">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="name" type="xsd:string" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="numberElement">
        <xsd:simpleContent>
            <xsd:extension base="tns:jsonnumbertype">
                <xsd:attribute name="name" type="xsd:string" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="booleanElement">
        <xsd:simpleContent>
            <xsd:extension base="xsd:boolean">
                <xsd:attribute name="name" type="xsd:string" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

</xsd:schema>
```

Appendix B. JSONx to JSON Stylesheet (not normative)

The following XSLT stylesheet takes a JSONx document as input and generates JSON. It is intended as a sample implementation, and makes no attempt to be well-behaved if the input is not well-defined.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:json="http://www.ibm.com/xmlns/prod/2009/jsonx">

    <xsl:output method="text" encoding="utf-8" indent="no"
      media-type="application/json"/>

    <xsl:template name="json:doNameAttr">
       <xsl:if test="local-name(..)!='array' and string-length(@name)>0">
          <xsl:value-of select="concat('&quot;', @name, '&quot;', ':')"/>
       </xsl:if>
    </xsl:template>

    <xsl:template match="json:object">
        <xsl:call-template name="json:doNameAttr"/>
        <xsl:text>{ </xsl:text>
        <xsl:for-each select="*">
           <xsl:apply-templates select="."/>
            <xsl:if test="position() != last()">
                <xsl:text>, </xsl:text>
            </xsl:if>
       </xsl:for-each>
       <xsl:text> }</xsl:text>
    </xsl:template>

    <xsl:template match="json:array">
        <xsl:call-template name="json:doNameAttr" />
        <xsl:text>[ </xsl:text>
        <xsl:for-each select="*">
            <xsl:apply-templates select="." />
            <xsl:if test="position() != last()">
                <xsl:text>, </xsl:text>
            </xsl:if>
        </xsl:for-each>
        <xsl:text> ]</xsl:text>
    </xsl:template>

    <xsl:template match="json:string">
        <xsl:call-template name="json:doNameAttr"/>
        <xsl:text>"</xsl:text>
        <!-- XXX Need to replace " with &amp;quot; -->
        <xsl:value-of select="normalize-space()"/>
        <xsl:text>"</xsl:text>
    </xsl:template>

    <xsl:template match="json:number">
       <xsl:call-template name="json:doNameAttr"/>
       <xsl:value-of select="normalize-space()"/>
    </xsl:template>

    <xsl:template match="json:boolean">
       <xsl:call-template name="json:doNameAttr"/>
       <xsl:value-of select="normalize-space()"/>
    </xsl:template>

    <xsl:template match="json:null">
        <xsl:call-template name="json:doNameAttr"/>
        <xsl:text>null</xsl:text>
    </xsl:template>

</xsl:stylesheet>
```

Authors' Addresses

Brien Muschett
IBM
8051 Congress Avenue
Boca Raton, FL 33487
USA

Phone: +1 561-862-2180
Email: muschett@us.ibm.com

Rich Salz
IBM
550 King Street
Littleton, MA 01460
USA

Phone: +1 978-899-2902
Email: rsalz@us.ibm.com
URI: https://www.ibm.com/developerworks/mydeveloperworks/blogs/soma/

Michael Schenker
IBM
555 Bailey Ave.
San Jose, CA 95141
USA

Phone: +1 408-463-3907
Email: mschenk@us.ibm.com