Lost in Translation:

Mappings between XML and JSON

David A. Clarke
(on behalf of Lattice Working Group)

PUNCH4NFDI – TA4 WP3

28 March 2022

UNIVERSITÄT BIELEFELD

Fakultät für Physik

Particles, Universe, NuClei and Hadrons for the NFDI

A consortium in the NFDI.

PUNCH 4NFDI

Foreward

This talk is an informal discussion of metadata structures `XML` and `JSON`. In particular we want to explain

- some `XML` basics
- some `JSON` basics
- how we use `XML` in lattice
- some difficulties mapping XML $\leftrightarrow$ JSON
- and what we've accomplished so far

Outline

XML and JSON

XML basics

An XML file is roughly a structured collection of tags.

```
<?xml version = "1.0" encoding = "UTF-8"?>

<!--
     XML documents have a tree structure starting at the "root"
     and branches to the "leaves". Sometimes also in the context
     people use jargon like "parent", "child", and "sibling".
-->

<note xmlns="http://www.w3.org/TR/html4/">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

- white space preserving
- flexible structures
- arguably not very readable

XML schemata

An advantage of XML: Built-in validation through schemata.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.w3.org/TR/html4/"
           xmlns="http://www.w3.org/TR/html4/"
           elementFormDefault="qualified">

<xs:element name="note">
<xs:complexType>
<xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>
```

Can check whether file validates against some schema:

```
xmllint --schema helloWorld.xsd helloWorld.xml --noout
```

```
helloWorld.xml validates
```

JSON basics

JSON file: collection of key-value pairs.

```
{
  "to" : "Tove",
  "from" : "Jani"
}
```

- No built-in schema validation[1].
- More readable for very simple structures.
- Easy to use with JavaScript.

[1]There exist solutions for this, e.g. JSON Schema .

Conversions

ILDG

Lattice calculation: Want estimator $\overline{X}$ for $\langle X \rangle$.

1. Generate gauge field configurations with MCMC.
2. Measure X on fields. Compute $\overline{X} = N_{conf}^{-1} \sum_i X_i$.

Strong motivations to reuse:

- Good signal can take $\mathcal{O}(1000)$ GPU-years and PB of storage.
- Can e.g. use configurations to compute $\overline{Y}$.

We combine efforts with International Lattice Data Grid (ILDG) .

- Already existing FAIR framework.
- Well known within lattice community.

`XML` in ILDG context

We expand on `QCDml` metadata schema. Hierarchy of metadata:

configuration $\subset$ ensemble $\subset$ campaign

- Thorough metadata schema implemented in XML.
- For now, preserve configuration and ensemble schemata.
- Connect with NFDI at campaign level.
- Conversions JSON $\leftrightarrow$ XML relevant here.

Partial QCDml configuration skeleton

```
<gaugeConfiguration>

    <crcCheckSum>
      <-- To check whether a configuration is damaged -->
    </crcCheckSum>

    <management>
      <-- Who made it, when it was made, revision history, etc. -->
    </management>

    <implementation>
        <machine> </machine>
        <code>    </code>
    </implementation>

    <algorithm>  </algorithm>

    <precision>  </precision>

    <markovStep>
        <markovChainURI>
          <-- Links configurations to ensemble -->
        </markovChainURI>
        <dataLFN>
          <-- Unique name for configuration  -->
        </dataLFN>
    </markovStep>

</gaugeConfiguration>
```

Python modules

Our first attempts at converting between these structures utilized Python modules like `json2xml` and `xmltodict`. Advantages:

- easy to use and read
- some already existing modules
- `requirements.txt`

But some care is needed:

- modules may not do what we want
- need to be sure modules are accessible

Minimalistic conversion script

```
import json, xmltodict, sys
from json2xml import json2xml
from json2xml.utils import readfromstring

xmlFile        = sys.argv[1]
out_xmljson    = 'XML_to_JSON.txt'
out_xmljsonxml = 'XML_to_JSON_to_XML.txt'

# We will turn this xml file into a dictionary. All the 'tags' in xml
# format will become 'keys' of the dictionary.
documentDict = xmltodict.parse( open(xmlFile).read() )

# Convert to JSON
documentJSON = json.dumps( documentDict, indent=4 )
outfile = open(out_xmljson,'w')
outfile.write(documentJSON)
outfile.close()

# Convert back to XML
documentXML  = json2xml.Json2xml(readfromstring(documentJSON),attr_type=False).
      to_xml()
outfile = open(out_xmljsonxml,'w')
outfile.write(documentXML)
outfile.close()
```

Results: Attributes

Original XML:

```
<xml>
  <t1 a="attrval"/>
  <t2 b="attrval">
    c2
  </t2>
</xml>
```

XML $\longrightarrow$ JSON $\longrightarrow$ XML:

```
<?xml version="1.0" ?>
<all>
  <xml>
    <t1>
      <key name="@a">attrval</key>
    </t1>
    <t2>
      <key name="@b">attrval</key>
      <key name="#text">c2</key>
    </t2>
  </xml>
</all>
```

Results: Ordering

Original XML:

```
<xml>
  <c>x1</c>
  <b>x2</b>
  <a>x3</a>
</xml>
```

XML $\longrightarrow$ JSON $\longrightarrow$ XML:

```
<?xml version="1.0" ?>
<all>
  <xml>
    <c>x1</c>
    <b>x2</b>
    <a>x3</a>
  </xml>
</all>
```

Results: Repeated tags

Original XML:

```
<xml>
  <a>x1</a>
  <a>x2</a>
  <b>x3</b>
  <a>x4</a>
</xml>
```

XML $\longrightarrow$ JSON $\longrightarrow$ XML:

```
<?xml version="1.0" ?>
<all>
  <xml>
    <a>
      <item>x1</item>
      <item>x2</item>
      <item>x4</item>
    </a>
    <b>x3</b>
  </xml>
</all>
```

Resolving ambiguities

```
{
  "xml": [
    { "a": "x1" },
    { "a": "x2" },
    { "b": "x3" },
    { "a": "x4" }
  ]
}
```

- Ordering and repeated tags: Use arrays [] and objects {}
- Nesting structure to match XML structure
- Try to avoid attributes

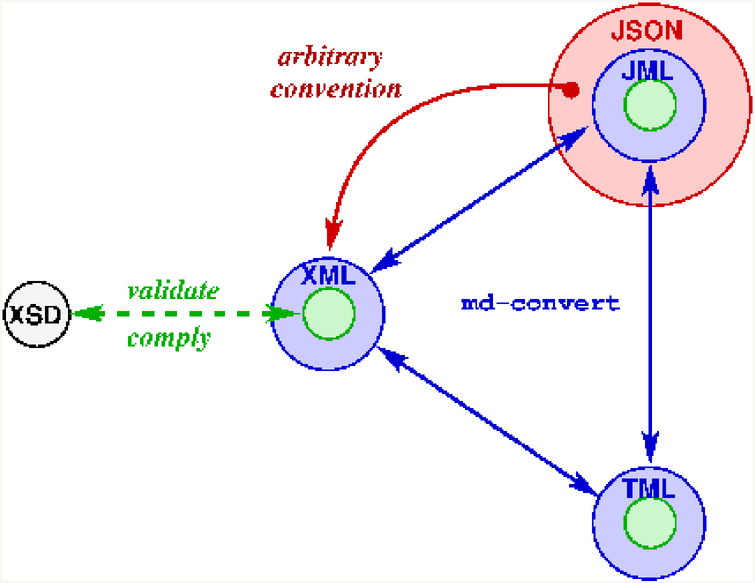
More sophisticated conversion

Nice Perl script `md-convert`. Features include but not limited to:

- Converts XML to JSON (JML), **reversibly**.
- Checks reversibility.
- Can validate the XML implementation.

```
./md-convert -o ${jsonfile} -oj1 -rev ${xmlfile}
```

Summary of mappings



Wrap Up

Summary

Efforts and plans so far:

- Expand on QCDml data schema.
- Requires XML $\leftrightarrow$ JSON conversions.
- Naive Python implementations fail reversibility.
- Working on more careful conversions.

Helpful links:

- 🔗XML intro , 🔗JSON intro
- 🔗ILDG information , 🔗QCDml documentation

Thanks for listening!