A simpler and shorter representation of XML data

inspired by Tcl

Version 2013-09-23

Agenda

- Who I am
- What is XML
- XML readability issue
- Alternatives to XML
- The search for a simpler way to present XML itself
- The SML proposal
- The sml conversion script
- Other scripts of interest
- SML dreams

Who I am

- Jean-François Larvoire, aka. JF, or Jeff
- Graduated from Ecole Centrale Paris in 1984
- Software developer since 1985 at HP, in Grenoble France, and Roseville and Sunnyvale California.
- Worked for several years on Tcl management scripts for Lustre storage clusters
- Now using more often Windows than Linux.
  Most examples for Windows.
- Contact: jf.larvoire@free.fr
  with sml in the subject

XML for Mummies

- A document standard published in 1998 by the W3C
- Structured text, representing a tree of data
- Syntax based on the older SGML standard
- Distinguishes text and <markup>
- The design goals were compatibility with SGML and HTML, versatility, ease of use (x4), simplicity (x2), conciseness (x2).
- XML 1.0 is most popular data exchange standard now
- XML 1.1 almost never used

Well formed XML

- **Nodes of the data tree are called elements**
- **Each XML document contains an XML header, and one root element**

```
<?xml version="1.0"?>
<greeting>Hello, XML world!</greeting>
```

- **Elements may contain inner elements, etc.**

```
<?xml version="1.0"?>
<world>
  <country>France</country>
  <country>USA
    <state>California</state>
    <state>Louisiana</state>
  </country>
</world>
```

Elements structure

- **Normal elements, with a content**

```
<name attrib1="value1" ...
attribN="valueN">content</name>
```

start tag **end tag**

The content is any combination of text and inner elements.

- **Empty elements**

```
<name attrib1="value1" ... attribN="valueN" />
```

- **Extra spaces and new lines between tokens are not significant**
- **Element names and attribute names must begin with a letter, and contain only letters, digits, and '.', '-', '_', ':'.**
- **Attribute values must be enclosed in 'quotes' or "double quotes", and may not contain '<', '&', nor their own limiter quote.**

Other notable features

- **Processing instructions**
  `<?xml version="1.0"?>`
- **Comments**
  `<!-- This is an XML comment -->`
- **CDATA sections**
  `<![CDATA[<sample>XML element</sample>]]>`
- **Entities**
  `<='<' >='>' &='&' &apos;="'" &quot;='"'`
  ~~`&eacute;='é' &euro;='€'`~~
- **Declarations**
  `<!ELEMENT book (title, body, supplements?)>`
  A valid XML document is one that is well-formed, AND where all elements respect the declared constraints.

Is XML simple?

- Goal #6 in the XML specification was: XML documents should be human-legible and reasonably clear.
- The XML specification is several thousand lines long.
- Writing a full-featured XML I/O library is a huge work… Fortunately good libraries are available now.
- In practice, small pieces of XML limited to a few lines are indeed simple to read by humans.
- In practice, large XML files are obscured by lots of markup. => Very difficult to read by humans

XML files that drove me crazy (1/2)

```
<?xml version="1.0" ?>
<cib admin_epoch="0" epoch="0" num_updates="0">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <attributes>
          <nvpair id="cib-bootstrap-options-symmetric-cluster" name="symmetric-cluster" value="true"/>
          <nvpair id="cib-bootstrap-options-no-quorum-policy" name="no-quorum-policy" value="stop"/>
          <nvpair id="cib-bootstrap-options-default-resource-stickiness" name="default-resource-stickiness" value="0"/>
          <nvpair id="cib-bootstrap-options-default-resource-failure-stickiness" name="default-resource-failure-stickiness" value="0"/>
          <nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled" value="true"/>
          <nvpair id="cib-bootstrap-options-stonith-action" name="stonith-action" value="reboot"/>
          <nvpair id="cib-bootstrap-options-startup-fencing" name="startup-fencing" value="true"/>
          <nvpair id="cib-bootstrap-options-stop-orphan-resources" name="stop-orphan-resources" value="true"/>
          <nvpair id="cib-bootstrap-options-stop-orphan-actions" name="stop-orphan-actions" value="true"/>
          <nvpair id="cib-bootstrap-options-remove-after-stop" name="remove-after-stop" value="false"/>
          <nvpair id="cib-bootstrap-options-short-resource-names" name="short-resource-names" value="true"/>
          <nvpair id="cib-bootstrap-options-transition-idle-timeout" name="transition-idle-timeout" value="5min"/>
          <nvpair id="cib-bootstrap-options-default-action-timeout" name="default-action-timeout" value="600s"/>
          <nvpair id="cib-bootstrap-options-is-managed-default" name="is-managed-default" value="true"/>
          <nvpair id="cib-bootstrap-options-cluster-delay" name="cluster-delay" value="60s"/>
          <nvpair id="cib-bootstrap-options-pe-error-series-max" name="pe-error-series-max" value="-1"/>
          <nvpair id="cib-bootstrap-options-pe-warn-series-max" name="pe-warn-series-max" value="-1"/>
          <nvpair id="cib-bootstrap-options-pe-input-series-max" name="pe-input-series-max" value="-1"/>
        </attributes>
      </cluster_property_set>
    </crm_config>
    <nodes/>
    <resources>
      <primitive class="ocf" id="ost1" provider="heartbeat" type="Filesystem">
        <operations>
          <op id="ost1_mon" interval="120s" name="monitor" timeout="60s"/>
        </operations>
        <instance_attributes id="ost1_inst_attr">
          <attributes>
            <nvpair id="ost1_attr_0" name="device" value="/etc/sfs/luns/lun8"/>
            <nvpair id="ost1_attr_1" name="directory" value="/mnt/ost1"/>
            <nvpair id="ost1_attr_2" name="fstype" value="lustre"/>
          </attributes>
        </instance_attributes>
      </primitive>
      <primitive class="ocf" id="ost2" provider="heartbeat" type="Filesystem">
        <operations>
          <op id="ost2_mon" interval="120s" name="monitor" timeout="60s"/>
        </operations>
        <instance_attributes id="ost2_inst_attr">
          <attributes>
            <nvpair id="ost2_attr_0" name="device" value="/etc/sfs/luns/lun10"/>
            <nvpair id="ost2_attr_1" name="directory" value="/mnt/ost2"/>
            <nvpair id="ost2_attr_2" name="fstype" value="lustre"/>
          </attributes>
        </instance_attributes>
      </primitive>
```

XML files that drove me crazy (2/2)

```
    <clone id="stonith_quincy3">
      <instance_attributes id="stonith_quincy3_inst_attr">
        <attributes>
          <nvpair id="stonith_quincy3_attr_1" name="clone_max" value="2"/>
          <nvpair id="stonith_quincy3_attr_2" name="clone_node_max" value="1"/>
        </attributes>
      </instance_attributes>
      <primitive class="stonith" id="stonith_hb_quincy3" provider="heartbeat" type="external/riloe">
        <operations>
          <op id="stonith_hb_quincy3_mon" interval="30s" name="monitor" prereq="nothing" timeout="20s"/>
          <op id="stonith_hb_quincy3_start" name="start" prereq="nothing" timeout="20s"/>
        </operations>
        <instance_attributes id="stonith_hb_quincy3_inst_attr">
          <attributes>
            <nvpair id="stonith_hb_quincy3_attr_2" name="hostlist" value="quincy3"/>
            <nvpair id="stonith_hb_quincy3_attr_3" name="ilo_hostname" value="192.168.16.153"/>
            <nvpair id="stonith_hb_quincy3_attr_4" name="ilo_user" value="jimi"/>
            <nvpair id="stonith_hb_quincy3_attr_5" name="ilo_password" value="secret:-)"/>
            <nvpair id="stonith_hb_quincy3_attr_6" name="ilo_can_reset" value="1"/>
            <nvpair id="stonith_hb_quincy3_attr_7" name="ilo_protocol" value="2.0"/>
            <nvpair id="stonith_hb_quincy3_attr_8" name="ilo_powerdown_method" value="off"/>
          </attributes>
        </instance_attributes>
      </primitive>
    </clone>
    <clone id="stonith_quincy4">
      <instance_attributes id="stonith_quincy4_inst_attr">
        <attributes>
          <nvpair id="stonith_quincy4_attr_1" name="clone_max" value="2"/>
          <nvpair id="stonith_quincy4_attr_2" name="clone_node_max" value="1"/>
        </attributes>
      </instance_attributes>
      <primitive class="stonith" id="stonith_hb_quincy4" provider="heartbeat" type="external/riloe">
        <operations>
          <op id="stonith_hb_quincy4_mon" interval="30s" name="monitor" prereq="nothing" timeout="20s"/>
          <op id="stonith_hb_quincy4_start" name="start" prereq="nothing" timeout="20s"/>
        </operations>
        <instance_attributes id="stonith_hb_quincy4_inst_attr">
          <attributes>
            <nvpair id="stonith_hb_quincy4_attr_2" name="hostlist" value="quincy4"/>
            <nvpair id="stonith_hb_quincy4_attr_3" name="ilo_hostname" value="192.168.16.154"/>
            <nvpair id="stonith_hb_quincy4_attr_4" name="ilo_user" value="jimi"/>
            <nvpair id="stonith_hb_quincy4_attr_5" name="ilo_password" value="secret:-)"/>
            <nvpair id="stonith_hb_quincy4_attr_6" name="ilo_can_reset" value="1"/>
            <nvpair id="stonith_hb_quincy4_attr_7" name="ilo_protocol" value="2.0"/>
            <nvpair id="stonith_hb_quincy4_attr_8" name="ilo_powerdown_method" value="off"/>
          </attributes>
        </instance_attributes>
      </primitive>
    </clone>
  </resources>
  <constraints>
    <rsc_location id="rsc_location_ost1" rsc="ost1">
      <rule id="prefered_location_ost1" score="100">
        <expression attribute="#uname" id="prefered_location_ost1_expr" operation="eq" value="quincy3"/>
      </rule>
    </rsc_location>
    <rsc_location id="rsc_location_ost2" rsc="ost2">
      <rule id="prefered_location_ost2" score="100">
        <expression attribute="#uname" id="prefered_location_ost2_expr" operation="eq" value="quincy4"/>
      </rule>
    </rsc_location>
  </constraints>
 </configuration>
 <status/>
</cib>
```

XML alternatives

- **Facing the same difficulty, many others have proposed alternative data formats.**
- **Ex: JSON**

```
"world": [
  { "country" = "France" },
  { "country" = "USA",
    "states" = [
      { "state" = "California" },
      { "state" = "Louisiana" }
    ]
  }
]
```

- **Pros: Considerably more legible; Trivial to parse in JavaScript… but not in other languages**
- **Cons: Incompatibility; Fragmentation; Limited Support**

Alternatives on the tcl.tk wiki

- **Xmlgen - Only designed to make it simple to generate XML, not as an alternative.**
- **TDL - Very similar to what I propose in many respects.**
  Pros: Strictly compatible with the Tcl syntax, contrary to what I propose.
  Cons: Not binary compatible with XML; Less human friendly syntax for text, cdata sections, comments, etc.

Do we have to give up XML compatibility?

- Fundamental equivalence of all structured text trees, starting with XML data files and Tcl programs.
- DOM
- Tcl as a member of the C family of languages (Java, PHP, etc)
- Is it possible to represent XML's DOM using a C-like syntax?
  - In a way 100% compatible with XML.
    (Allowing 100% fidelity back and forth conversions)
  - With as few changes as possible.

The SML solution

XML (from a Google Earth .kml file)	SML (generated by the sml script)

```
<?xml version="1.0" encoding="UTF-8"?>
<kml>
 <Folder>
  <name>Take off zones in the Alps</name>
  <open>1</open>
  <Folder>
   <name>Drome</name>
   <visibility>0</visibility>
   <Placemark>
    <description>Take off</description>
    <name>Mont Rachas</name>
    <LookAt>
     <longitude>5.0116666667</longitude>
     <latitude>44.8355</latitude>
     <range>4000</range>
     <tilt>45</tilt>
     <heading>0</heading>
    </LookAt>
   </Placemark>
  </Folder>
 </Folder>
</kml>
```

```
?xml version="1.0" encoding="UTF-8“
kml {
 Folder {
  name "Take off zones in the Alps"
  open 1
  Folder {
   name Drome
   visibility 0
   Placemark {
    description "Take off“
    name "Mont Rachas“
    LookAt {
     longitude 5.0116666667
     latitude 44.8355
     range 4000
     tilt 45
     heading 0
    }
   }
  }
 }
}
```

SML Elements

- Elements normally end at the end of the line.
- They continue on the next line if there's a trailing '\'.
- They also continue if there's an unmatched "quotes" or {braces} block.
- Multiple elements on the same line must be separated by a ';'.

Pros: A natural match for most modern XML files, which usually have one XML terminal element per line; Same as Tcl instructions.

Cons: ?

SML Attributes

- **The syntax for attributes is the same as for XML. Including the rules for using quotes, avoiding ‘<' and '&', and escaping using entity chars.**

```
bird species='eagle' note="Found here &amp; there"
```

Pros: Conversion straightforward; Readable as it is.

Cons: Not the same as Tcl’s string quoting and escaping rules.

SML Content Data

- **The content data are normally inside a {curly braces} block.**
  ```
  tag {content}
  ```
- **The content text is between "quotes".**
  Escape '\' and "" with a '\'.
  ```
  tag {"Some text with an \" and";{an inner element} }
  ```
- **If there are no further child elements embedded in contents (i.e. it's only text), the braces can be omitted.**
  ```
  tag "Some text with an \" but no inner element"
  ```
- **Furthermore, if the text does not contain blanks, "", '=', ';', '#', '{', '}', '<', '>', nor a trailing '\', the quotes around the text can be omitted too. (ie. It cannot be confused with an attribute or a comment or any kind of SML markup.)**
  ```
  tag 3.1415926535
  ```

Pros: Removes as much syntactic clutter as possible, with a result looking a lot like Tcl.

Other types oF SML markup

- This is a `?Processing instruction` .
  (The final '?' in XML is removed in SML.)
- This is a `!Declaration` . (Ex: a `!doctype definition`)
- This is a `#-- Comment block, ending with two dashes --` .
- Simplified case for a `# One-line comment .`
- This is a `<[[ Cdata section ]]>`. Initial \n discarded.

Pros: Conversion straightforward; Minimizes clutter.

Cons: Alternatives closer to Tcl syntax are possible. Ex:

```
{
  An indented Cdata section
}
```

The sml conversion script

- Converts XML to SML, then back to XML without any change.
- I've been using it for several years.
- Tested on a 1MB corpus of XML files from various sources.

Demo

SML script status

- About 3000 lines of Tcl, half of which are an independent debugging library.
- Performance: It converts about 10 KB/s of data on a 2 GHz machine. => Should be rewritten in C if high perf needed.
- Known limitation as of 2013-09-23:
  - The converted files use the local operating system line endings .

The show script

- Displays a file tree as SML.
- Each file or directory is an SML element.
- Directories contain inner elements that represent files and subdirectories.
- File contents are displayed as text if possible, else are dumped in hexadecimal.
- Several modes of operation, with standard or experimental SML, and a simplified output mode that's most readable.

Demo

The spath virtual script

- A thought experiment that gives some insight on the power of the SML concept.
- The xpath script for extracting XML data using XPATH
- Writing a full-featured script able to use XPATH to extract data from SML files would be very difficult.
- Yet this can be done in a single line of code:

```
sml | xpath %*
```

- Powerful in combination with the show script

Demo

SML files sizes

- The total size of the converted files is 12% smaller than my original XML test files.
- Among big files, that reduction goes from 4% for a file with lots of large CDATA elements, to 17% for a file with deeply nested elements.
- Even after zipping the two full sets of samples, the SML files archive is 2% smaller than the XML files archive. Microsoft may like to use this to shoehorn a few more Office .docx or .pptx documents into Windows phones. ☺

Using the SML format for exporting tcl data

The SML data format is a natural format for exporting any kind of Tcl data:

- **SML looks a lot like Tcl itself. (Particularly when not using attributes.)**
- **In simple cases (without attributes), SML can often be parsed by the Tcl interpreter directly as Tcl lists of lists.**
- **You get for free the compatibility with any outside tool that supports XML.**

Using SML for network protocols

- The savings potential is even better in XML-based network protocols, such as SOAP.
- Adapting existing XML-based protocols to use SML instead would be very easy, and increase bandwidth considerably.
- Creating new ad-hoc SML-based protocols would be easy too, and packet analysis would be much easier!

Side note about Tcl name spaces

The convergence of XML element trees, file system trees, and Tcl or C++ variable name spaces, leads me to think that the natural syntax for name spaces should not be:
::namespace::subspace::variable
nor:
namespace.subspace.variable
but instead:
/namespace/subspace/variable

An SML parsing library?

- I never had to write one…
- In simple cases (often) SML just looks like Tcl lists, which the Tcl interpreter can digest directly.
- In complex cases (rarely), it was too easy to use the sml script to generate XML, then the TclDOM package to parse it. So I never took the time to write a full-fledged parsing library.
- Of course, ~~for the sake of the art~~ for performance reasons, it would be nice to write one in C or C++, using APIs similar to XML parsing libs.
- The simplest is probably to modify TclDOM or tDOM packages to parse either format. Any volunteer?

An SML generation library?

- I never had to write one…
- Simple SML files are straightforward to generate in Tcl.
- Actually I do have one library… for PowerShell

```
Put-Dir dirname [attributes] [script_block]
Put-EndDir dirname
Put-Value name [-nameWidth N] [attributes] value
```

  Generates either XML or SML
- Can easily be rewritten in Tcl
- Better still, for Tcl, modify TclDOM or tDOM packages to generate either XML or SML. Any volunteer?

Next Steps

- Let people experiment with the tools, and give feedback about the syntax, and the possible alternatives.
  Scripts available at http://jf.larvoire.free.fr/progs/
- Is there any error or inconsistency that remains, preventing full XML compatibility in some case? If so tell me: jf.larvoire@free.fr
- If interest grows, work with interested people to freeze a standard, and create a simple SML generation and parsing library.
- If interest grows even further, work with the TclDOM and tDOM developers to see if this could be added to their libraries as an alternative XML encoding format, for both input and output.
- Consider using for your editing of complex XML files.
- Consider using for storing data in a friendly XML-compatible way.