Using XML in Internet Protocols

Tim Bray
Distinguished Engineer
Director of Web Technologies
Sun Microsystems

Using XML in Internet Protocols

Tim Bray
Distinguished Engineer
Director of Web Technologies
Sun Microsystems

Agenda

- Should you use XML?
- Should you invent a new XML language?
- If you’re inventing a new XML language, how do you maximize your chances of success?

Should You Use XML? Other options:

- Hardwired binary
- ASN.1
- Plain text
- JSON
- XML

Hardwired Binary: Issues

- Compact.
- (Potentially) high-performance parsing.
- Architecture-dependence.
- Severe debugging pain.

Example: IPV? packet headers

Use Hardwired Binary If:

- You're *way* down the protocol stack.
- But even then, be nervous.

ASN.1: Issues

- Compact.
- IETF tradition.
- Lousy tools.
- Debugging hell.
- No community outside the IETF & ITU.
- Only metadata is data type.

Example: SNMP

Use ASN.1 If:

- You have to talk to other IETF stuff that’s locked in.

Plain Text: Issues

- The simplest possible option is often the best.
- Pretty efficient.
- Fits well with server-side Internet (Unix) culture.
- Watch out for I18n.
- Watch out for extensibility.

Example: HTTP

Use Plain Text If:

- ... you possibly can.

JSON: Example vs. XML

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON: Issues

- Superb browser integration.
- Knows about lists, tuples, hashes.
- Maps directly to programming-language structures.
- Hard-wired to UTF-8 (in theory).
- Awkward for deeply-nested or “document”-style structures.
- Watch out for extensibility.
- Browser security issues.

Example: Google Maps mashups

Use JSON If:

- You’re shipping structs and tuples around from program to program.
- You expect to implement client software in-browser.
- The expected lifetime of the data is short.
- It isn’t text-heavy.

XML: Issues

- Tons of excellent open-source tools.
- Programmers love XPath.
- Decent extensibility.
- I18n is nailed.
- Handles “document” structures well.
- Verbose & ugly.
- Doesn’t map naturally to programming-language structures.
- DOM API is programmer-hostile.

Use XML If:

- Your data is document-flavored.
- You’re worried about i18n.
- You’re worried about extensibility.
- You’re worried about reusability.

So, you’re going to use XML...

Inventing New XML Languages:

- Time-consuming.
- Bureaucratic.
- Difficult.
- Unpleasant.
- Includes complex software development as a sub-task.
- Usually fails.

Inventing New XML Languages:

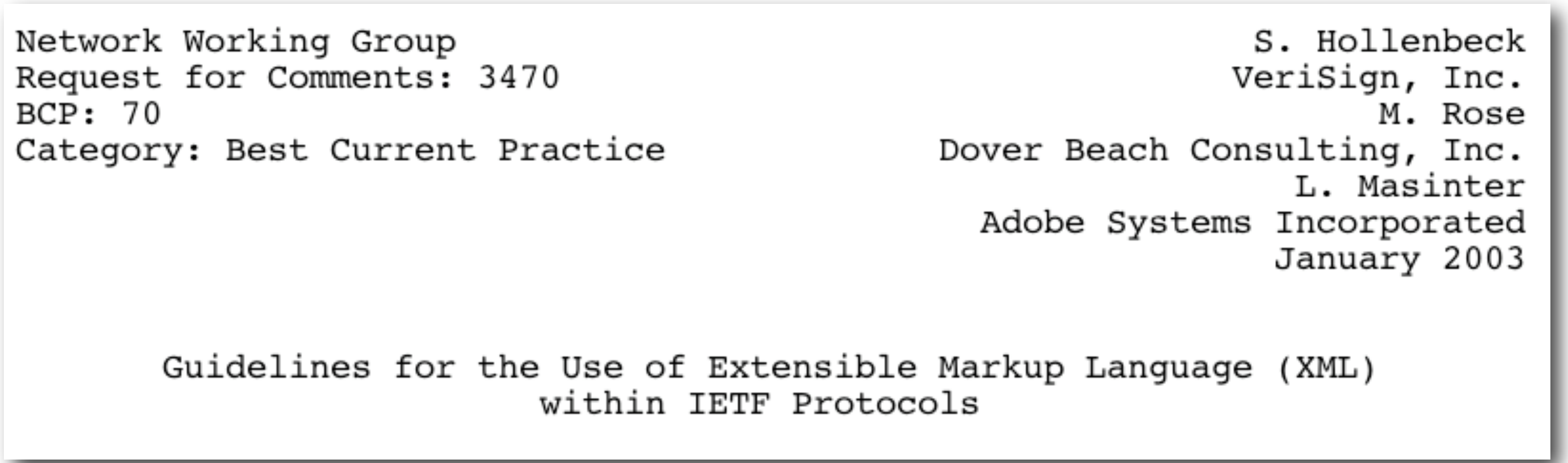
- Time-consuming.
- Bureaucratic.
- Difficult.
- Unpleasant.
- Includes complex software development as a sub-task.
- Usually fails.

... so try not to!

Some Good XML Languages

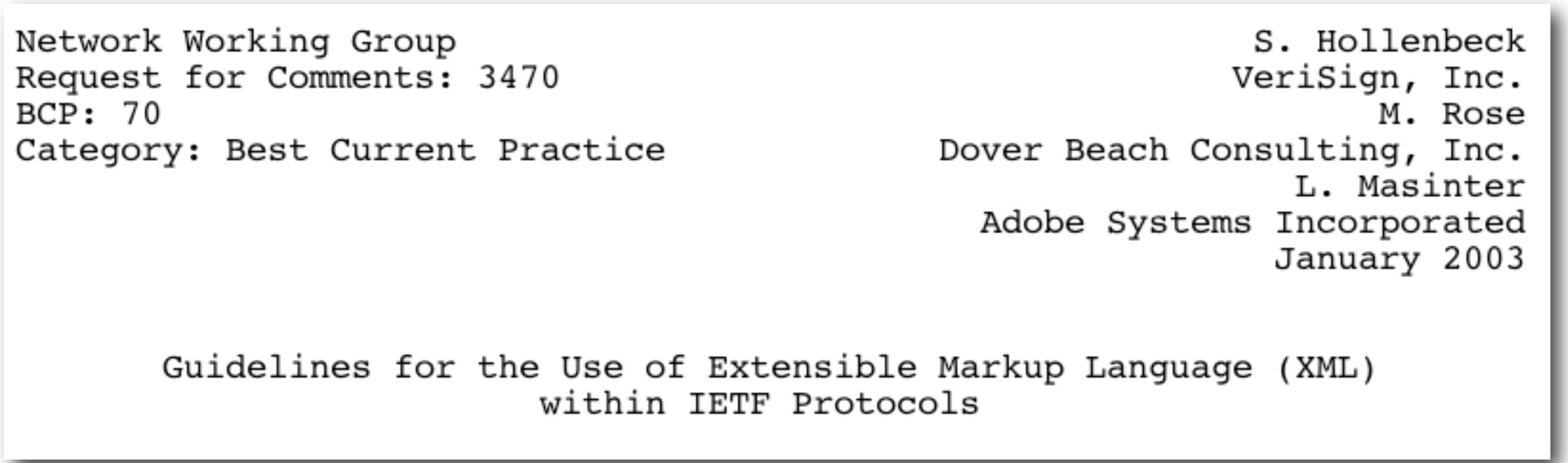
- XHTML
- DocBook
- ODF
- Atom
- XMPP
- UBL
- RDF

So, you’re making your own language...



```
Network Working Group                                      S. Hollenbeck
Request for Comments: 3470                                VeriSign, Inc.
BCP: 70                                                          M. Rose
Category: Best Current Practice             Dover Beach Consulting, Inc.
                                                             L. Masinter
                                               Adobe Systems Incorporated
                                                             January 2003


       Guidelines for the Use of Extensible Markup Language (XML)
                          within IETF Protocols
```



```
Network Working Group                                      S. Hollenbeck
Request for Comments: 3470                                VeriSign, Inc.
BCP: 70                                                          M. Rose
Category: Best Current Practice             Dover Beach Consulting, Inc.
                                                              L. Masinter
                                             Adobe Systems Incorporated
                                                            January 2003


      Guidelines for the Use of Extensible Markup Language (XML)
                         within IETF Protocols
```

Design Issue: Semantics

- What does “Age” mean?
- What does “Version” mean?
- What does “Person” mean?
- What does “Update” mean?
- What does “Creator” mean?

Design Issue: Model vs. Syntax

“What matters is getting the data model right. The syntax is ephemeral.”

“The bits on the wire are the only reality.”

Design Issue: Minimalism vs. Completeness

"Let's solve the whole problem."

"Minimum progress required to declare victory."

Design Issue: Specification Tools

- Human-readable prose.
- Examples.
- Validator.
- Schema.

But, first: Know Your Audience

Why specs matter

Most developers are morons, and the rest are assholes. I have at various times counted myself in both groups, so I can say this with the utmost confidence.

-Mark Pilgrim: http://diveintomark.org/archives/2004/08/16/specs

Design Issue: Specification Tools

- Human-readable prose.
- Examples.
- Validator.
- Schema.

Design Issue: Specification Tools

- Human-readable prose. ← Most important
- Examples. ← Very important
- Validator. ← Very important
- Schema. ← Nice to have

XML Schema Language Options

- DTD
- XSD (W3C XML Schemas)
- RelaxNG
- Schematron

Document Type Definitions (DTDs)

- Constrain only what elements/attributes can appear, and where.
- Don’t say much about content.
- Allow the definition use of “Entities”, macros of zero arguments. Don’t use them!
- Past their sell-by date.

W3C XML Schemas (XSD)

- Hard to understand, hard to implement, hard to interoperate.
- No underlying formalism.
- Limited in the set of markup idioms they can define.
- Includes (in “Part 2”) a usable set of primitive data types: Integers, floats, dates, URIs, and so on.
- One of the reasons why the SOA/WS-* project is sinking.

RelaxNG

- Based on the hedge-automaton formalism.
- Written in XML, or a non-XML Compact Syntax.
- Good human-readability.
- Can specify a very wide range of markup idioms.
- Can use XSD Part 2 base datatypes.
- Validators only available in Java and C.
- For a good example, see RFC4287.
- ISO 19757-2.

Schematron

- Based on XPath.
- Assertions with associated error/success messages.
- Excellent for checking for specific error conditions or anomalies.
- Not really a language-specification tool.
- Several implementations.
- ISO 19757-3.

XML Extensibility: Three Options

- No changes.
- Must-Understand policy (e.g. as in SOAP).
- Must-Ignore policy (e.g. as in Atom).

XML Internationalization

- “An XML document knows what encoding it’s in.” -Larry Wall
- In an ideal world, everything would be in UTF-8.
- In the real world, people don’t understand this stuff and probably shouldn’t have to.
- XML makes this survivable in many circumstances... with most tools, they can suck up their Shift-JIS or Big5 or whatever and it’ll quite possibly Just Work.

XML Security and Signatures

```
<a b="1" c="1"/> <a
                   c='1'
                   b='1'></a>
```

- Shouldn’t these two have the same signature?
- XML Canonicalization is the solution.
- Unfortunately, it’s also a problem.
- XML DigSig says how to apply a signature to c14n-ized XML.
- Or, you could just sign the bag-o’-bits.

The Semantic Web

- The RDF view: Everything’s a graph of 3-tuple assertions: Resource/Property/Value.
- R, P, and V can each be a URI. Value can be a URI or a literal.
- Assertions can be resources.
- The RDF/XML serialization is ugly and annoying.
- Semantic Web project sees a bright future of operations on the Universal graph, once it’s built, so they’d like to use RDF/XML for everything.

Thank You!

Tim.Bray@sun.com
tbray.org/ongoing/