What to do about ANY?

OSCAL Model Meeting

March 5, 2021

Ways to provide extensibility

Of three approaches we have formulated so far, today's discussion is about the (so-called) ANY strategy.

Extension by restriction

- Provide generic tags with qualifiers for semantic labeling

```
<prop name="whiz"
  ns="xyz.org">bang</prop>
```

- We have `prop`, `annotation`, `part` (at least) available for use

ANY

- Exploit features of (respective) schema languages to provide hooks for developers
- Effectively, these can provide for "mix-ins" to mingle with OSCAL data

New models

- Write your own OSCAL metaschema
- Leverage OSCAL tooling
- Reuse OSCAL building blocks

ANY in schemas

In XSD, `<xs:any namespace="##other" maxOccurs="unbounded"/>`

> We would use this to permit elements in “any other” (non-OSCAL) namespace in certain contexts.

In JSON Schema, `"additionalProperties": true`

What this looks like

Use case: To embed bibliographic data acquired from extant data (in some other format) – we want to save the entire record, while exposing normalized data derived from it. This gives us traceability and fidelity even while it enables capturing the good stuff without having to cast or map it.

We can do this with ANY inside a catch-all `biblio` element.

```
<resource xmlns="http://csrc.nist.gov/ns/oscal/1.0" uuid="...">
   <title>Syntax for the Digital Object Identifier - ANSI/NISO Z39.84-2005 (R2010)</title>
   <rlink href="…Digital%20Object%20Identifier.pdf"/>
   <citation>
       <text>ANSI/NISO Z39.84-2005 (R2010), Syntax for the Digital Object Identifier.
…Digital%20Object%20Identifier.pdf</text>
       <biblio>
        <std xmlns="http://www.niso-sts.org"><std-ref type="undated">ANSI/NISO Z39.84-2005
(R2010)</std-ref>, <title>Syntax for the Digital Object Identifier</title>. <ext-link
xlink:href="…Digital%20Object%20Identifier.pdf" ext-link-
type="uri">…Digital%20Object%20Identifier.pdf</ext-link></std>
       </biblio>
    </citation>
</resource>
```

In (W3C) XML, we rely on namespaces to indicate embedded foreign XML vocabularies.

`xmlns` works as a switch to show a processor where we are not using OSCAL.

Possible uses

- Enhance OSCAL with local/private/specialized application semantics
  - Extension by restriction also supports this
- Go beyond limits of “markup-multiline”
  - For free-form paragraph, list and simple table contents, OSCAL has only minimalist Markdown-compatible tagging
  - Go beyond these limits by mixing in (X)HTML or an alternative documentary format (e.g., NISO STS, DITA, StratML)
- Embed (not just link) non-OSCAL evidentiary or testimonial data into OSCAL

Potential challenges

What should OSCAL XML<->JSON converters do with non-OSCAL content?

Pass through in some fashion?

Provide hooks for custom converter code?

Provide option to drop?

How do we address expectations around use of ANY features?

(Given that any ANY is a black box to OSCAL more or less by definition)

Current status

OSCAL models are developed and produced using NIST Metaschema technology, which provides documentation and tooling along with modeling capabilities.

We have now implemented ANY in the metaschema processing runtime.

Now we have support for it, the question comes up, whether and where to use ANY in OSCAL?

Where we allow ANY

Group

Control (at any level)

Part (at any level)

Parameter

- Parameter Guidance
- Parameter Selection

Resource

- Resource bibliographic citation

In models to date, we have provided ANY in certain places – now due for review

Alternatives to ANY

OSCAL features we already have in place:

- *Abstract generics* mentioned above – `prop`, `part` etc.
- Links – anything can be aligned with links
- Back matter resources – as descriptions of out of line content, these can also provide links, fallback links, discoverability metadata, and even (binary) contents (base-64 encoded)

Do we want ANY?