
Topic Maps – Data Model, XTM Syntax

Washington DC, November 12, 2004

Resolution of final issues

Agenda for the day

- **Markup in occurrences and variants**
 - proposal for how to represent this
- **Discussion of remaining issues**

Markup in topic maps



Previous agreements
How to get it in there
Consequences

xml-data-representation

- **Amsterdam decisions**
 - XML markup is not allowed in base names
 - it *is* allowed in variants and internal occurrences
- **No decisions were made regarding representation**

How to specify this in TMDM/XTM

- **XTM describes a mapping from the XML Infoset to TMDM**
- **So the contents of <resourceData> elements are already represented as an XML Infoset to begin with**
 - the only difference is that now we allow elements in here
- **The XTM specification will say that the contents are to be turned into a string, following Canonical XML**
 - this will give us a string preserving the original XML fragment
 - including all namespace information etc
 - it also ensures that lexically different XML fragments will be equal inside the topic map
- **The result is stored in the existing [value] property**
 - variant.[value]
 - occurrence.[value]
- **Note: we explain it this way in the spec, but it doesn't have to be *implemented* this way**

So how do we know that this is XML?

- **Given that [value] already exists and contains a string, how can we distinguish the two cases below?**
 - `<resourceData>this is not XML</resourceData>`
 - `<resourceData>this is XML</resourceData>`
- **Proposal: we add a [type] property, with the following possible values**
 - XML
 - string
 - (list to be left open, so other types can be added)
- **XTM deserialization will set the [type] property correctly**

Ok, but how do we identify types?

- We use PSIs, of course
- The [type] property contains a locator
 - xsd:any XML
 - xsd:string string

Yeah, but aren't locators also a type?

- **Well, yes, they are**
- **So we could unify [resource] and [value]**
- **This would mean that**
 - occurrence.[resource] and variant.[resource] are removed
 - another type is added to the list: tmdm:locator
 - XTM deserialization deals with this
- **Again, implementations do *not* have to represent it this way, but it does simplify the explanation of topic maps**
- **Equality rule:**
 - [type] and [value] must both be equal
- **Problem:**
 - occurrence.[value] and variant.[value] can now hold either a string or a locator item
 - this makes the typing of this property pretty awkward

Uh, why do we have locators, anyway?

- **Good question**
- **Originally: to be able to support Hytime locators from HyTM**
- **However, now there is no standard syntax supporting anything but URIs**
 - nor is TMQL and TMCL likely to support it
 - nor does any known software support it
 - nor does any known user want it
- **In short, why don't we ditch the locators item entirely?**
 - the locator item type goes
 - locator notations go
 - all properties that hold locators become string properties
 - the PSI for the locator type becomes `xsd:anyUri`, instead of `tmdm:locator`
 - typing of `*.[value]` becomes uniformly “string”
 - XTM can now represent any TMDM instance (which it couldn't before)
 - TMDM becomes significantly shorter

Decisions

- **Change [type] in the proposal to [datatype]**
 - “type” is overloaded in TMDM, also more specific
- **Otherwise: this is fine**
- **Canonical XML processing**
 - make sure it supports multiple elements without a wrapper parent
 - strip out comments and PIs
 - make sure all namespace declarations *inside resourceData* are preserved
- **foo prefix is defined *and* used for an element (preserved)**
 - `<resourceData><foo:address ...>...</foo:address></resourceData>`
 - `<resourceData><bar:address ...>...</bar:address></resourceData>`
- **foo prefix is defined *outside resourceData* and *not* used for an element or attribute name (not preserved)**
 - `<topicMap xmlns:foo="...">....<resourceData><bar>foo:something`

Using <resourceData> as a wrapper?

- **[value] if we use a wrapper *and* preserve all namespaces in scope**
 - `<resourceData xmlns="http://www.topicmaps.org/xtm/1.0/" xmlns:xlink="http://www.w3.org/1999/xlink"><a/></resourceData>`
 - `<resourceData xmlns="http://www.topicmaps.org/xtm/1.0/" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:foo="http://www.example.org/foo"><foo:address>...</foo:address></resourceData>`
- **[value] if don't use a wrapper, and we don't preserve unused decls**
 - `<a/>`
 - `<foo:address xmlns:foo="http://www.example.org/foo">...</foo:address>`
- **We try the second approach in the first draft**
 - this will be reviewed to see if it's acceptable

Remaining issues



,

Add a *datatype* attribute on `<resourceData>`?

- This would let us put other types of data into topic maps besides just strings
- **1. Should we add the attribute?**
 - yes, and this means we adopt a “type follows value” policy
- **2. Should it be required?**
 - it can't be, because that would be incompatible with XTM 1.0
 - default type is `xsd:string`
- **3. What happens if you leave it out and use XML markup?**
 - it's an error, because default is string, and anyway it's useful to know before you start parsing that the content is complex XML
- **4. Can we then lose `<resourceRef/>`?**
 - no, that would mean that XTM 1.0 documents are not valid 1.1 documents
 - this is the recommended representation
- **5. What happens if you put `<resourceData datatype=xsd:anyUri>...?`**
 - it's equivalent to `<resourceRef/>`, and we consider `<resourceRef/>` a shorthand in the same way that `<instanceOf>` is a shorthand for an association

More fun with datatypes

- **6. What happens if you put datatype=xsd:number and “five”?**
 - it is an error
- **7. What happens if you put datatype=foo:complexNumber?**
 - the type information is preserved in TMDM
 - validation of the string representation at the XTM level is not *required*
 - XTM processors can support plug-in type validators, but we do not standardize how these work, only how the types are identified
- **8. How do we keep the XTM representation manageable?**
 - that is, how do we keep the syntax for datatyping concise?
 - this cannot be delegated to TMCL; it has to be an XTM concept
 - we want it to be purely syntactic, and *not* represented in TMDM, only the result should be in TMDM, not the mechanism itself
 - we need to avoid risk of future merging changing interpretation of the data or causing conflicts

One alternative short syntax for datatyping

- **Tie the data type to the occurrence type**
 - `<topicMap><occurrenceDataType datatype="xsd:date"><topicRef xlink:href="#birthdate"/></occurrenceDataType>`
 - `<topic ...>...<occurrence><instanceOf><topicRef xlink:href="#birthdate"/></instanceOf><resourceData>...</>`
- **Considerations**
 - topic identification is complex, so the occurrence may use a different identifier from the declaration, and we may only discover that they are the same *after* we have read the occurrence
 - we can work around this by saying that unless the occurrence uses *exactly* the same topic identification the datatyping behaviour does not kick in
 - however, this means `<topicRef>` behaves differently here from elsewhere
 - easy to ensure datatyping of occurrences is consistent
 - will lead to arguments of the form “why isn't this part of TMCL?”, but the answer is that this is just a shorthand



Another alternative short syntax for datatyping

- **Define elements for the data types**

- `<topicMap><dataTypeElement
datatype="xsd:date">xsd:date</dataTypeElement>`
- `<topic ...>...<occurrence><instanceOf><topicRef
xlink:href="#birthdate"/></instanceOf><xsd:date>...`

- **Considerations**

- avoids the topic identification issue
- is obviously just syntactical
- namespace issues
 - if no prefix is used the element will default to the XTM namespace
 - requires processors to know what namespaces are in effect
- looks a lot like RDF/XML
- one alternative is to not declare the element, but just use namespace information to resolve the PSI of the datatype
 - loses some validation of the XML input
- another alternative is to define that a particular namespace is a datatype namespace (`dataTypeNamespace` instead of `dataTypeElement`)

Conclusion to question #8

- We don't like any of our proposals so far
- Not convinced that we really need to do this
- However, don't want to close the door to further proposals
- The next draft will *not* have a shorthand syntax in it

Even more fun with datatypes

- **9. How is the string value validated against the declared type?**
 - for the XTM-recognized data types the string value *must* match the lexical representation of that data type
 - for other data types this is not required, but we allow XTM processors to be extended with validation for other data types (via undefined means)
 - **note:** we *could* let the type default to the datatype in which the string is a legal lexical representation, but we concluded that this may lead to data integrity problems
- **10. What are the allowed lexical representations of date types?**
 - specs say CCYY-MM-DD, and if you don't like this that's your problem, but you *can* choose to use a different data type that has the representation you like

Fun, fun, fun

- **11. Do we normalize the string representation of non-string values?**
 - ie, does “05” turn into “5”?
 - important for merging/duplicate suppression
 - XML Schema, part 2, does not specify normalized representations
 - implementations which represent numbers as numbers (ie optimized) *will* normalize, and they cannot avoid it except by keeping the string around
 - there are three choices for each data type
 - normalization is required (now: string, XML)
 - normalization not required, but allowed (now: URIs)
 - normalization forbidden
 - we know most implementations will not normalize strings, so we remove the string normalization requirement
 - we are not really normalizing XML, just turning it into a string in a defined way, which we need to do anyway
 - we *stop* saying that URIs may be normalized, and don't specify any normalization for them
 - so effectively we choose “normalization forbidden” for all three types
 - normalization is allowed, but it is considered authoring, and not part of normal deserialization

Fun, part 12

- **12. What's the list of datatypes defined in TMDM?**
 - xsd:string, xsd:any (XML), xsd:anyUri, at least
 - what the reason to include a data type in this list?
 - it gives validation for values of this type
 - comparison, operations, etc all belong in TMQL, and so are not reasons
 - conclusion: only the three basic types (because they have to be there, and this keeps complexity of implementation down)

Not really all that fun any more

- **13. Is there a mechanism for typing external resources?**
 - at present: no
 - what's needed to do this is a set of PSIs for resource types plus for making the assertion that a particular resource is of a particular resource type
 - applications will want to consider the same resource to belong to different resource types in different circumstances
 - such a mechanism does not belong in ISO 13250 because
 - you can create it yourself,
 - it's way too complicated to build into the standard
- **14. What is the datatype extensibility mechanism?**
 - the ability to use values in the *datatype* attribute in XTM other than the three predefined types
 - we do not provide any mechanism whatsoever for making statements about the datatypes used
 - the reason is that there is nothing to say about the type, since we don't have any operations on them other than string equality testing

Note on TMCL and TMQL

- **TMCL can be used to verify that**
 - values are valid according to their declared data types, and that
 - their declared data types conform to the TMCL schema declarations
- **TMCL does *not* modify the topic map in any way, such as by changing the data types of values based on the schema**
 - ie, nothing like the XML Schema post-validation infoset
- **TMCL may require support for certain datatypes**
- **TMQL will for certain datatypes support**
 - ordering and comparison of values of this type, and
 - operations on values of this type (such as addition)

Example of datatyped occurrence data

```
<!-- note: this is the only allowed way; there is no shorthand -->
<topicMap xmlns:xsd="http://something...">
  <topic id="tosca">
    <occurrence>
      <instanceOf><topicRef xlink:href="#premiere-date"/></instanceOf>
      <resourceData datatype="xsd:date">1900-01-14</resourceData>
    </occurrence>
  </topic>
</topicMap>
```

The term “reification”

- **There are two cases**
 - the general topic-subject relationship
 - the relationship between a topic and a topic map construct whose subject is being represented by the topic
- **Agreed solution: we add a NOTE explaining that the use of the term “reification” in TMDM is not to be confused with the use of the in philosophy**

association item	topic item
association	topic
relationship	→ subject

Two remaining XTM issues

- **xm-pubid**
 - XTM 1.0 uses "`-//TopicMaps.Org//DTD XML Topic Map (XTM) 1.0//EN`"
 - we want one, primarily because it would look odd not to have one, and we will use the standard syntax for FPIs defined in ISO standards
 - within that constraint we try to make it as similar to the previous one as possible (but changing “XML Topic Map” to “XML Topic Maps”)
- **xm-topicref-xmwrap**
 - we solve this by removing support for embedding `<topicMap>` elements in other XML vocabularies
 - the text in the draft supporting this will be removed



Instructions

- **Authors are being instructed to prepare a new draft for ballot as FCD**



XML Schema for XTM 1.1

- Ann's proposal