Comments from Bill Barnhill on “xdu-rdf-model-v8.pdf”

General Comments
The document is well-written and thorough, but seems to describe too much. I'd recommend splitting this document into the following separate documents (with rough outlines):

- “The XDI RDF Model”
  - Enhancements to plain RDF model (new section)
    - XDI RDF document similar to RDF A-Box (i.e. assertions)
    - XDI RDF dictionary similar to RDF T-Box (i.e. Terminology)
    - No blank nodes
    - Statement as address and vice versa
    - Other benefits
    - Further enhancements found in “XDI Dictionaries”
  - How XRIs are leveraged
    - Global Context Symbols
    - Direct Concatenation
    - Cross References
  - Basic Structure and Addressing
  - Contexts, Context References, and Context Descriptors
  - The four core predicates
  - Abstract API for XDI RDF Model Create/Read/Update/Delete operations
  - Java binding for API
  - Ruby binding for API
  - Python binding for API
- “X3 Serialization for XDI RDF”
  - Basic concepts
  - Two versions:
    - Simple (newline delimited list of XRIs, each XRI asserts one or more statements)
    - Full (whitespace indented, what's now X3D)
  - Examples
  - Abstract API for XDI RDF serialization and deserialization
  - Java binding for API for XDI RDF serialization and deserialization
  - Ruby binding for API for XDI RDF serialization and deserialization
- Python binding for API for XDI RDF serialization and deserialization

- **“REST-Based API for XDI RDF”**
  - XDI Messages
    - The four atomic REST operations
    - The $set aggregate operation
    - The $do operation and creating custom operations

- **“Asynchronous API for XDI RDF”**
  - Document I am working on
  - Basic concepts (still rough)...
    - Higher bandwidth than REST API, but is asynchronous and document-centric. Also allows for use of link contracts within API and long running operations
    - An operation is defined as data interchange pattern between 2 or more data authorities
    - Performing an operation consists of...
      - Requesting authority creating and populating a request XDI RDF graph under its authority (e.g. =example/+originated.request/1 ). The root of the XDI graph represents the request, which is expected to have at least two predicates:
        - $response, which either has a $data predicate with an empty object, or is a link to another XRI through a link contract with write privileges. If $data the response is written into this graph by the interchange target authority (multiple authorities mean multiple versions). If a link then the response is written into the linked graph.
        - $data, which is the request payload. The expected content of the XDI graph which is the object of this predicate is determined by the service's XDI dictionary, which serves as XDI RDF's version of WSDL. This service dictionary will have one or more XRIs of the form +/$has/+XXX, which indicates that the object of the $data predicate must be the subject of one $is$a predicate for each $XXX. An example is the following dictionary, which would mean the $data object would need to have the properties satisfying membership in +person, +employee, and +supervisor. (example TBD, but it has those three as objects of +/$has)
          - A service's dictionary would be found under the interchange target authority but might link to a shared dictionary elsewhere. Having the XRIs for service dictionaries be of a consistent form aids discovery. Such a form could be having the subject as the service authority (e.g. @example+service*stockquote) and + as the predicate (e.g. @example+service*stockquote/+).
      - Requesting authority uses pre-negotiated link contract to create new XRI under interchange target authority. This XRI has as subject the root of the XDI graph describing the service (e.g. @example2+service*stockquote) and +request as the predicate. The object is an XRI cross-reference to the request context under the requesting authority (e.g. (=example/+originated.request/1 )). The complete example created XRI would be @example2+service*stockquote/+request/(=example/+originated.request/1). The subject and predicate of this XRI are essentially the
'service endpoint' of this service.

- The execution of the service is triggered by creation of this XRI. When a response is generated it is written under the requesting authorities' +response/+data (e.g. =example/+originated.request/1/+response/+data), or under the XRI that is linked to from the +response.

- Note that link contract policies could be used to enable pipelining by using the +response objects as subsequent request objects.

- SOAP correspondences:
  - Message exchange pattern is async two one-way MEPs
  - WS-Addressing 'To' functionality accomplished by combination of +response XRIs
  - WS-Addressing 'From' functionality accomplished by request XRI
  - Aspects of WS-Security accomplished by Link Contracts
  - SOAP Body equivalent to request's +data object
  - SOAP headers equivalent to request's other objects

- “Known XDI RDF Implementations”
  - X3 Converter
  - X3 Validation service

- “XDI RDF Versioning”
  - Version logging
  - Version snapshots
  - Subject Versioning
  - Predicate Versioning
  - Examples
    - Create new XDI subject
    - Change =drummond's e-mail address
    - Add an i-number synonym
    - Change =drummond's e-mail address again
    - Delete =Drummond's name
    - Complete example

- “XDI Link Contracts”
  - What is a Link Contract
  - Benefits over existing authorization mechanisms
  - Examples

- “XDI Dictionaries”
  - What is an XDI Dictionary?
- XDI representation of an ontology
  - Enhancements over plain RDF ontology
    - XRI-enabled
    - Access controlled via link contracts
    - Versioned
  - The XDI RDF type dictionary
  - The XDI RDF Metagraph model

**Section Specific Comments**

This is organized as an outline corresponding to 1st level headers in v8 document. It would be easier to address comments if document had line numbers, so if possible I'd recommend future drafts add those.

**Comments on 'About the Proposed XRI 3.0 Syntax'**

The discussion on direct concatenation doesn't mention scheme, but you cannot append an XRI if it is absolute and contains the scheme. Seems better to say 'The authority and path of any valid XRI can be appended to the path of a second valid XRI and the result is also a valid XRI. A segment delimiter of '/' is not assumed and if not present the appended XRI is a new subsegment on the last path segment of the transformed XRI.'

**Comments on 'The XDI RDF Model'**

For the purpose of these comments I'm going to call the XRI syntax of XDI RDF in this draft the n-segment syntax. This corresponds to A/pred.about.A/obj.of.pred.about.A/pred.about.obj.of.pred.about.A/obj.of.pred.about.A and so on.

This diverges from my original concept of X3 and it seems to both re-introduce the perils of striping and to re-introduce blank nodes. The blanks node seem to be re-introduced by the 3rd bullet, *If the XRI in the third segment is relative...*). One of the beautiful things about using XRI for RDF is that blank nodes aren't needed anymore. N-segment syntax could also seem confusing for newcomers to read I suspect, specifically with regards to meaning of fourth non-empty segment as there are two intuitive meanings that are at odds with each other: (a) the 4th segment is a predicate about the object described in the 3rd segment, and (b) the 4th segment is a predicate about the statement which is composed of segments 1-3. (a) is the correct way according the v8 draft.

An alternative syntax similar to the original X3 concept is to use cross-references where you would use more than three segments in the n-segment syntax. Since this has at most 3 segments I call this the 3-segment syntax. This syntax may not be as easily read, but seems more precise and does not introduce blank nodes.

As an example the following table contrasts the N3 way against the n-segment syntax and the 3-segment syntax:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3</td>
<td>&lt;s1&gt; &lt;p1&gt; &lt;o1&gt; .</td>
</tr>
</tbody>
</table>
One problem with the 3-Segment syntax is that the N-Segment syntax uses cross-references for reification. This means 3-Segment syntax has to have a different notation for a subject which is the statement itself rather than the object of the statement. The 3-Segment notation for this is a cross-reference within a cross-reference: ((s/p/o)). So a subject of (s/p/o) asserts s/p/o and starts a new statement whose subject is o, while a subject of ((s/p/o)) asserts s/p/o and starts a new statement whose subject is the statement s/p/o. For example to say that =Drummond is author of the statement =Bill.Barnhill is a contributor to the resource represented by @example we would use the XRI:

((((@example/+dc+contributor/=Bill.Barnhill))/+dc+author/=Drummond).

Another proposed component of 3-Segment syntax is the ability to designate that the subject of a subsequent statement is the subject or predicate of the previous statement rather than the default of the object of the previous statement. This is done by appending a subsegment of either +subject, +predicate, or if wishing to be redundant +object onto the subject cross-reference.

For example to say that =Bill.Barnhill is a contributor to @example while =Drummond is the author we would use the XRI:

(@example/+dc+contributor/=Bill.Barnhill)+subject/+dc+author/=Drummond

3-Segment syntax also allows precedence to be explicit. For example consider the following 3-Segment XRI: (s1/p1/o1)/p2/(o2/p3/o3). To express this in n-Segment syntax we would need two XRIs: (a) s1/p1/o1/p2/o3 and (b) o2/p3/o3. This allows us to nest an entire XDI graph into one XRI as long as there is a traversable path in the graph from any resource to any other resource in the graph.

Also note that in 3-Segment syntax the third segment can be empty (e.g. @example/+dc+author/). This denotes the list of XRIs for objects in the graph that satisfy that predicate. In some cases this list will only contain one member. When there are no satisfying objects this will be an empty list.

An equivalent to the n-segment context syntax of // may be needed for 3-segment syntax, but the same functionality seems available by using cross-references. As an example

=drummond@(http://oasis-open.org)/+email/(=drummond@cordance/+email)

Variables also seem easier in 3-segment syntax. To describe a basic graph pattern to search for all non-red cars and return 3 column result of XRI, +V*Car, +V*Color we would use the XRI:

(+V*Car/$is/+car)/$hasColor/(+V*Color/$is.not/+red)

This may be Further study is needed to determine if this is possible with one n-segment syntax XRI.

Several parts of this section would be better moved to separate documents, allowing this document to be tightly focused. The sub-sections recommended for moving are the one on REST operations, the Type dictionary, and the sub-section on link contracts. See general comments previous for a recommended set of documents with a rough outline.
Comments on 'X3 Serialization Format'

The v8 draft specifies 3 serialization formats: Standard X3, X3W, and X3D. X3D is closest to the format as proposed in the original X3, but is cleaner at the expense of requiring whitespace-aware editing and parsing, which seems acceptable. Also the acronym X3D already stands for a 3D visualization standard from W3C that is the proposed successor to VRML. That X3D has decent if not wide adoption. X3 seems to use brackets to eliminate need for whitespace. X3W seems to be a separate format for corner-case usages.

As a result I propose streamlining this down to 2 formats, X3/Simple and X3/Full (naming TBD). X3/Simple is a newline delimited list of XRI's following 3-Segment syntax. X3/Full is what is X3D in v8 of draft, but limited to a depth of 3 to match 3-Segment restriction.

The mechanism of context descriptors is not sitting well with me, but I haven't got a good replacement for them so until I can put forth a better mechanism I'll be ok with context descriptors. I also haven't pinpointed what it is about context descriptors that's bothering me, which is another reason not to propose changing them at this time.

This also seems to belong in a separate document, as outlined in the general comments section previously.

Comments on 'Example XDI RDF Documents'

Rather than have a single example section I'd recommend removing the versioning and messaging sections to other documents and have each document give examples of what that document covers.