DML: XML CONVERSION

Contact: Wiley Black (WBlack@optics.arizona.edu)

Version: V3.1 Draft

CONTENTS

Overview....1

DML Compared to XML....1

XML-DML Translation....2

- Boolean Encoding....3
- Date Time Encoding....3
- Opaque Data Encoding....3
- Extended Primitives....3

OVERVIEW

XML provides a "human-readable" encoding format that is suitable to many, many purposes. XML contains a rich, yet simple structure which can represent almost any data that a computer can understand or transmit. While it can represent an arbitrary structure, it can't necessarily do so efficiently. That is where binary formats come into play.

Data Markup Language provides the structural power and flexibility of XML while providing a strict, tight representation that is suitable for use in both high and low-resource environments, and makes use of both in order to provide maximum capabilities.

DML COMPARED TO XML

Some of the key differences between DML and XML include:

- DML is a strict binary encoding.
- XML is a less restrictive (i.e. whitespace) textual encoding.
- DML is strongly typed, excluding opaque data.
- XML can be typed, but often is not.

- No schema mechanism currently defined by DML. Although an XML schema can be referenced in a DML translation using XMLRoot.
- Translations are unique to DML and provide the conversion table necessary to translate DML to XML.
- Namespaces are allowed as part of node names in DML, but are not specifically processed in any way by DML. Most definitions in the base dml translation table have a DML: prefix that can be removed through use of a namespace definition after conversion to XML.
- XML Root provides a facility for specifying the namespaces that are necessary in XML but unnecessary in DML.

XML-DML TRANSLATION

Many parts of DML are completely analogous to XML and have clear counterparts in both. Key points in the translation process are:

- The DML Translation document and/or inline identification provides the mapping between DML Identifiers and XML element and attributes names.
- All attributes and elements that are included in the XMLRoot container of the DML Translation document should be merged with the top-level container of the XML.
- Container elements translate to XML elements.
- Empty containers must be carried between XML and DML as their presence and multiplicity can carry information.
- Attributes translate to XML attributes attached to their immediate container element.
- Signed and unsigned integer and floating-point primitives translate to strings.
- XML elements can contain both attributes and text content. In DML, these can be mapped to a DML Container element with attributes followed by a XML:CDATA string primitive. See note below.

The DML Translation information is not ordinarily included in XML conversion, however it can be included as a comment within the XML if so desired.

There is one structural mapping that is indirect between DML and XML. An example is given as:

```
<Example-XML-Element Attribute=”Present”>
      Textual Content
</Example-XML-Element>
```

The above requires 3 DML nodes for representation:

- DML Container Element to wrap it all. The container has name “Example-XML-Element”.
- DML Attribute for “Attribute”, a string primitive with value “Present”.
- XML:CDATA string primitive for “Textual Content”.

During the conversion from DML to XML the XML:CDATA string primitive can be automatically recognized and replaced a standard XML CDATA section. A more optimal approach will scan the CDATA and determine whether it qualifies as CDATA or PCDATA (parsed). If it qualifies as PCDATA without alteration, then the string primitive’s content can be written without a CDATA wrapper.

During conversion from XML to DML, any text content that cannot be represented by a DML string primitive must be wrapped in an XML:CDATA node.

BOOLEAN ENCODING

In conversion to XML, “true” and “false” are valid representations as per XML Schema Part 2: Datatypes Second Edition (http://www.w3.org/TR/xmlschema-2/).

DATE TIME ENCODING

When converted to XML, a date-time primitive should be represented in ISO 8601 format. A tool which presents an XML-Like tree view of the DML may elect to display the date and time in any format.

OPAQUE DATA ENCODING

Opaque data located in an attribute cannot be represented properly in XML, although a base-64 string is the closest analogy.

Opaque data in an element can be converted to and from XML as PCDATA enclosed in an element whose name is determined in the same way as DML container element names (see below). For example, a single DML data element named “Message” might be translated to/from:

```
<Message encoding=”base64”>

VGhpcyBpcyB0aGUgaW5mb3JtYXRpb24gY29udGVudC4=

</Message>
```

Or it could be translated as:

```
<Message>This is the information content.</Message>
```

Since the translator will not ordinarily have any information about the format of the message, base64 encoding is recommended in the absence of interpreter-provided knowledge and the encoding=”base64” attribute is also recommended in the XML representation. No matching encoding attribute is present or necessary in the DML representation.

The element name “Message” is the name of the element.

EXTENDED PRIMITIVES

The extended primitives may or may not have clear translation to XML. The base-10 floating-point type can be converted to a string, although precision questions may arise. Arrays and matrices could be converted into a comma-separated textual representation. Extended primitives which are not interpreted by a reader must be handled as opaque data, and the same is true for conversion to and from XML.