**DTD/XSD (XML Validation) Orientation (3/30/2018)**

**DTD and XSD defined:**

A DTD is a collection of rules that defines a type of XML document. It includes things like element names, what attributes and children each element can have, and what order elements are in. It is used to check that a particular XML file meets the definition for its type, a process called validation. For example, we use a DTD to validate EAD finding aids so we know that they comply with the EAD standard.

An XSD file is a DTD that is encoded using the XML schema for DTDs. It allows more information to be included, like how many times an element must or can appear and restrictions on the content of an element, such as controlled vocabularies or how dates should be formatted.

**How to validate XML with a DTD:**

Use an XML processor to validate an XML file against a DTD. It will compare the XML file to all the rules in the DTD and give an error message for each way the XML document deviates from the rules. If it correctly follows all the rules, the XML document is considered valid.

You can do this in XML Blueprint. First, open an XML document and associate it with the schema (Schema - Associate XML Document with DTD…). Then validate it (XML - Validate). We use xmllint to do validation from the command line as part of the aip-finish\_script.

**Organization of our XSDs:**

There are three XSD files, one for each namespace: master.xsd, dc.xsd, and premis.xsd. The master.xsd has the main structure and also imports the other two, so you only need to associate an XML file with master.xsd and all three are applied during validation.

Within each XSD file, there are three main structures: <xs:element>, <xs:complexType>, and <xs:simpleType>.

* <xs:element> gives rules for an element: its name, children, etc.
* <xs:complexType> give rules for the order of an element's children.
* <xs:simpleType> gives rules for restrictions on an element's value.

<xs:complexType> and <xs:simpleType> can be children of <xs:element>. Alternatively, to repeat something for multiple <xs:element>'s or to make the document easier to read, <xs:complexType> and <xs:simpleType> can be put at the same level as <xs:element>. Then when an <xs:element> has a type attribute that matches the name of a <xs:complexType> or <xs:simpleType> sibling, the rules for that type are applied to that <xs:element>. (See example 3 below)

To compare it to XSLT, <xs:element> works similar to a root template (although there can be more than one sibling <xs:element>), <xs:complexType> and <xs:simpleType> work similar to match templates, and <xs:element> with a type attribute works similar to apply-templates.

During validation, elements are expected to be in the same order as they are in the DTD. Unless otherwise noted (with minOccurs and maxOccurs), elements are expected to be in the XML only once. There should only be one <xs:element> for each element name in the XML.

**Elements used in our XSD**

Elements may have other permitted children and attributes. This document only describes ones we currently use. In the list below, elements are a child of the element they are indented under.

* **xs:schema**: occurs only once in a document and contains all other elements.
  + **xs:import:** location of other XSD documents to use rules from.
  + **xs:element:** has rules (in attributes and/or child elements) for a particular element.
  + **xs:complexType:** defines children of an element. Can also be a child of <xs:element>.
    - **xs:choice:** XML must match one of its children, which are <xs:sequence>'s.
    - **xs:sequence:** Lists <xs:element>'s in a required order. One way to represent hierarchy is for these <xs:element>'s to have <xs:complexType> children, which have <xs:sequence> children, and so on. (see example 4).
  + **xs:simpleType:** defines restrictions on an element. Can also be a child of <xs:element>.
    - **xs:restriction:** Container for defining restrictions on an <xs:element>'s content.
      * **xs:enumeration:** a value that is permitted (repeat for each value).
      * **xs:minLength:** the minimum number of characters (min 1 = not empty).
      * **xs:pattern:** simplified regex that must match the whole value of the element.
        + [0-9]{#} means there must be # number of digits.
        + (text).+ means that it starts with "text", followed by anything else.
        + pattern|pattern means it must match one of the patterns.

**Attributes used in our XSD**

* **base:** data type, e.g. xs:anyURI, xs:integer, and xs:string.
* **elementFormDefault="qualified":** need to use prefixes for namespaces.
* **maxOccurs:** maximum times an element may be repeated. Use "unbounded' if unlimited.
* **minOccurs:** minimum times an element may be repeated. Use "0" if optional.
* **name:** name of the element.
* **namespace:** namespace that an imported document is in.
* **ref:** reference to an <xs:complexType> or <xs:simpleType> being imported from another XSD.
* **schemaLocation:** location of a document to import, relative to the location of the current XSD.
* **targetNamespace:** namespace of the current XSD document.
* **type:** data type, e.g. xs:string, or reference to name of an <xs:complexType> or <xs:simpleType>.
* **value:** definition of a restriction.
* **xmlns:x:** gives uri for namespace represented with prefix "x".

What attributes can be used with each element:

* **xs:complexType, xs:simpleType:** name
* **xs:element:** maxOccurs, minOccurs, name, ref, type (only use type if no <xs:restriction>)
* **xs:enumeration, xs:minLength, and xs:pattern:** value
* **xs:import:** namespace, schemaLocation
* **xs:restriction:** base
* **xs:schema:** elementFormDefault, targetNamespace, xmlns

**Examples:**

1. **An element with 2 children: one URL and one that is text.**

<xs:element name="objectIdentifier">

<xs:complexType>

<xs:sequence>

<xs:element name="objectIdentifierType" type="xs:anyURI"/>

<xs:element name="objectIdentifierValue" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

1. **An element with a restriction: it is a URL for rightsstatements.org or creativecommons.org.**

<xs:element name="rights">

<xs:simpleType>

<xs:restriction base="xs:anyURI">

<xs:pattern value="(http://rightsstatements.org/vocab/).+|(https://creativecommons.org/licenses/).+" />

</xs:restriction>

</xs:simpleType>

</xs:element>

1. **An element that uses type to call the rules from a <xs:simpleType> and a <xs:complexType>.**

<xs:complexType name="objectCharacteristics">

<xs:sequence>

<xs:element name="fixity" type="premis:fixity" minOccurs="0" maxOccurs="unbounded" />

*(other elements in the sequence go here)*

</xs:sequence>

</xs:complexType>

<xs:complexType name="fixity">

<xs:sequence>

<xs:element name="messageDigestAlgorithm">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="MD5" />

<xs:enumeration value="SHA-256" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="messageDigest" type="premis:hasString" />

<xs:element name="messageDigestOriginator" type="premis:hasString" minOccurs="0" />

</xs:sequence>

</xs:complexType>

<xs:simpleType name="hasString">

<xs:restriction base="xs:string">

<xs:minLength value="1" />

</xs:restriction>

</xs:simpleType>

1. **An element that nests <xs:element> and <xs:complexType> or <xs:simpleType> to define the children's children.**

<xs:element name="format">

<xs:sequence>

<xs:element name="formatDesignation">

<xs:complexType>

<xs:sequence>

<xs:element name="formatName" type="premis:hasString" />

<xs:element name="formatVersion" type="premis:hasString" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="formatRegistry" minOccurs="0">

<xs:complexType>

<xs:sequence>

<xs:element name="formatRegistryName" type="premis:hasString" />

<xs:element name="formatRegistryKey" type="premis:hasString" />

<xs:element name="formatRegistryRole" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="specification" />

<xs:enumeration value="validation profile" />

</xs:restriction>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="formatNote" type="premis:hasString" maxOccurs="unbounded" />

</xs:sequence>

</xs:element>