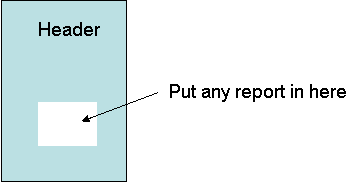
# Embedding Various Components into an XML Component

Roger L. Costello

November 2010

## Problem Statement

Suppose you have a header document and you have various reports that you would like to drop into the header.  The reports are unique and have their own schema.  How do you design the header schema and the report schemas so that the various reports can be dropped into the header?



## Example

Here is a sample header:

|  |
| --- |
| <Header xmlns="<http://www.example.org/header>">          <Author>John Doe</Author>        </Header> |

Here is a sample report:

|  |
| --- |
| <Report xmlns="http://www.example.org/report>         <Title>My report on XYZ</Title> </Report> |

We want to be able to put the report inside the header, at the bottom:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <Header xmlns="http://www.example.org/header">          <Author>John Doe</Author>      <Report xmlns="http://www.example.org/report">         <Title>My report on XYZ</Title>   </Report>  </Header> |

Below are four approaches to designing the header and report schemas so that reports can be dropped into the header.

## Approach #1: Use the <any> Element

Design the header schema to expect unforeseen components to be inserted at the bottom of the <Header> element. This is accomplished using the <any> element:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/header"  elementFormDefault="qualified">     <xs:element name="Header">  <xs:complexType>  <xs:sequence>  <xs:element name="Author" type="xs:string" />  <xs:any processContents="strict" />  </xs:sequence>  </xs:complexType>  </xs:element>   </xs:schema> |

Design report schemas completely independent of the header schema:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/report"  elementFormDefault="qualified">     <xs:element name="Report">  <xs:complexType>  <xs:sequence>  <xs:element name="Title" type="xs:string" />  </xs:sequence>  </xs:complexType>  </xs:element>   </xs:schema> |

An instance document author simply drops the desired report into the header:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <Header xmlns="http://www.example.org/header"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.example.org/header header.xsd  http://www.example.org/report report.xsd">    <Author>John Doe</Author>  <Report xmlns="http://www.example.org/report">  <Title>My report on XYZ</Title>  </Report>   </Header> |

Notice that the value of schemaLocation is two pairs – one for the header schema and one for the report schema.

## Approach #2: Use Abstract Element and Element Substitution

Design the header schema with an element that must be replaced (substituted). This is accomplished using an element that has a type which is abstract:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/header"  xmlns="http://www.example.org/header"  elementFormDefault="qualified">     <xs:element name="Header">  <xs:complexType>  <xs:sequence>  <xs:element name="Author" type="xs:string" />  <xs:element ref="Report" />  </xs:sequence>  </xs:complexType>  </xs:element>    <xs:element name="Report" type="Report-type" />    <xs:complexType name="Report-type" abstract="true"/>   </xs:schema> |

Note that the Report element has an abstract type. Thus, the Report element must be replaced (substituted).

The report schema contains a Report element that replaces (substitutes) the header schema's Report element:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/report"   xmlns="http://www.example.org/report"  xmlns:hdr="http://www.example.org/header"  elementFormDefault="qualified">    <xs:import namespace="http://www.example.org/header"  schemaLocation="header.xsd"/>    <xs:element name="Report" type="Report-type" substitutionGroup="hdr:Report"/>    <xs:complexType name="Report-type">  <xs:complexContent>  <xs:extension base="hdr:Report-type">  <xs:sequence>  <xs:element name="Title" type="xs:string"/>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>   </xs:schema> |

In XML instance documents the header's Report element is replaced with the Report element from the report schema:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <Header xmlns="http://www.example.org/header"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.example.org/header report.xsd">    <Author>John Doe</Author>  <Report xmlns="http://www.example.org/report">  <Title>My report on XYZ</Title>  </Report>   </Header> |

## Approach #3: Use Type Substitution

Design the header schema with the anticipation that the <Header> element's content may be extended or restricted. Do this by defining the <Header> element's type as a global complexType:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/header"   xmlns="http://www.example.org/header"  elementFormDefault="qualified">   <xs:element name="Header" type="Header-type"/>   <xs:complexType name="Header-type">  <xs:sequence>  <xs:element name="Author" type="xs:string"/>  </xs:sequence>  </xs:complexType>  </xs:schema> |

Design the report schema with a complexType that extends Header-type:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/report"   xmlns="http://www.example.org/report"  xmlns:hdr="http://www.example.org/header"  elementFormDefault="qualified">   <xs:import namespace="http://www.example.org/header"  schemaLocation="header.xsd" />    <xs:complexType name="Report-type">  <xs:complexContent>  <xs:extension base="hdr:Header-type">  <xs:sequence>  <xs:element name="Report">  <xs:complexType>  <xs:sequence>  <xs:element name="Title" type="xs:string" />  </xs:sequence>  </xs:complexType>  </xs:element>  </xs:sequence>  </xs:extension>  </xs:complexContent>  </xs:complexType>  </xs:schema> |

The XML instance document must specify the <Header> element's type:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <Header xmlns="http://www.example.org/header"  xmlns:rpt="http://www.example.org/report"  xsi:type="rpt:Report-type"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.example.org/header header.xsd  http://www.example.org/report report.xsd">    <Author>John Doe</Author>  <Report xmlns="http://www.example.org/report">  <Title>My report on XYZ</Title>  </Report>   </Header> |

Notice xsi:type="rpt:Report-type" on the <Header> element.

## Approach #4: Use NVDL

Design the header schema and the report schemas completely independent of each other. The header schema does not need to anticipate that unforeseen components will be dropped into it.

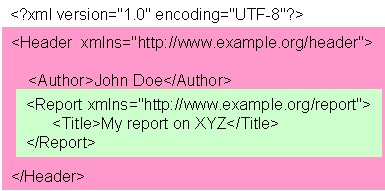
Here is the simple header schema:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/header"  elementFormDefault="qualified">     <xs:element name="Header">  <xs:complexType>  <xs:sequence>  <xs:element name="Author" type="xs:string" />  </xs:sequence>  </xs:complexType>  </xs:element>   </xs:schema> |

Here is the simple report schema:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  targetNamespace="http://www.example.org/report"  elementFormDefault="qualified">     <xs:element name="Report">  <xs:complexType>  <xs:sequence>  <xs:element name="Title" type="xs:string" />  </xs:sequence>  </xs:complexType>  </xs:element>   </xs:schema> |

Here is an XML instance document in which a Report component is embedded inside the Header component:



It is a "compound document." It consists of two components—the Header component and the Report component. An NVDL script specifies how to partition the compound document into its components and how to validate each component:

|  |
| --- |
| <?xml version="1.0"?> <rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0">    <namespace ns="http://www.example.org/header">  <validate schema="header.xsd"/>  </namespace>   <namespace ns="http://www.example.org/report">  <validate schema="report.xsd"/>  </namespace>  </rules> |

Read as: validate the component in the <http://www.example.org/header> namespace against header.xsd and validate the component in the <http://www.example.org/report> namespace against report.xsd.

See my NVDL tutorial for more info: <http://www.xfront.com/nvdl/>

## Analysis of the 4 Approaches

Approach 1 uses the <any> element:

|  |
| --- |
| <xs:any processContents="strict" /> |

This means that any element can be inserted at that position.

Approach 2 uses a tandem of element plus abstract type:

|  |
| --- |
| <xs:element name="Report" type="Report-type" />  <xs:complexType name="Report-type" abstract="true"/> |

Approach 2 is nearly equivalent to Approach 1. In Approach 2 the abstract Report element can be replaced (substituted) by any element that has a type which derives from Report-type.

Approach 3 uses type substitution. The instance document author must specify the <Header> element's type:

|  |
| --- |
| xsi:type="rpt:Report-type" |

With Approach 4 instance documents are assembled from arbitrary schemas, without change to the schemas, or *a priori* integration of the schemas. The information about how to validate each part of the XML instance document is declaratively specified in a standard XML vocabulary (NVDL):

|  |
| --- |
| <?xml version="1.0"?> <rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0">    <namespace ns="http://www.example.org/header">  <validate schema="header.xsd"/>  </namespace>   <namespace ns="http://www.example.org/report">  <validate schema="report.xsd"/>  </namespace>  </rules> |