Requirements for transclusion in DocBook

09 December 2010

This version:
: http://docbook.org/docs/transclusion-requirements/2010-12-09/

Latest version:
: http://docbook.org/docs/transclusion-requirements/

Author:
: Jirka Kosek, `<jirka@kosek.cz>`

Table of Contents

UC-1: Shared strings 1
UC-2: Shared strings in attribute values 3
UC-3: Conditional shared text 3
UC-4: Modularized documents 4
UC-5: Repeated transclusion in one “master” document 5
UC-6: Transclusion of foreign content 5
Evaluation of current technologies 6

This document summarizes uses-cases for transclusion of content in DocBook documents. This document should help DocBook TC in deciding whether existing standards like XInclude are sufficient for the task or whether DocBook specific mechanism should be designed.

Note

DocBook TC has not yet decided whether DocBook specific mechanism is really required or not. But draft proposal of such transclusion mechanism is available at http://docbook.org/docs/transclusion/.

UC-1: Shared strings

Many technical publication use repeating boilerplate text for things like product names or product versions. It is useful if such text is defined only once per document (or set of documents) and then just referenced. This approach prevents typos and makes updates and rebranding of content very easy.

Current solutions

Entities

This use-case can be solved by using internal entities[1]. This is well supported in processing tools and to some extent in authoring tools. However entities are somewhat hard to maintain when you use modular documents at the same time. Also entities and their references are not part of XDM so they do not survive XSLT processing which makes various “document massaging” tasks challenging.

[1] http://www.w3.org/TR/REC-xml/#sec-internal-ent

Example 1. Using entities for shared strings

```
<!ENTITY product-version "3.14">
<!ENTITY product-name "FooWiz">
<!ENTITY corp-name "ACME Inc.">

…
<para>The latest version of <application>&product-name;</application>
from &corp-name; is &product-version;.</para>
```

XInclude

XInclude[2] when combined with proper XPointer schema can be used for transclusion of single element or even only its text content. In this setup there can be separate file with definition of shared texts which are then reused in other documents.

Example 2. Definition file for further XInclude referencing

```
<article xmlns="http://docbook.org/ns/docbook" version="5.0">
  <title>Shared texts</title>
  <para><phrase xml:id="product-version">3.14</phrase></para>
  <para><phrase xml:id="product-name">FooWiz</phrase></para>
  <para><phrase xml:id="corp-name">ACME Inc.</phrase></para>
</article>
```

The problem is that support for different XPointer schemes varies between implementations. Almost all implementations support referencing to element specified by its ID as shown in the following example.

Example 3. Using XIncludes for shared texts

```
<… xmlns:xi="http://www.w3.org/2001/XInclude">
…
  <para>The latest version of <application><xi:include href="shared-texts.xml" xpointer=▶
"product-name"/></application>
    from <xi:include href="shared-texts.xml" xpointer="corp-name"/>
    is <xi:include href="shared-texts.xml" xpointer="product-version"/>.</para>
…
```

This solution has two problems. The actual reference to shared texts is very verbose because each `xi:include` element contains reference to the file with shared text definitions. Second problem is that such reference doesn't include only text of referenced element but whole element including `xml:id` attribute. This leads to excessive markup and duplicated IDs in the composed document.

Solution is to use more advanced XPointer scheme. This leads to the more arcane markup, support in tools is very bad, but there are no duplicated IDs and excessive markup.

[2] http://www.w3.org/TR/xinclude/

Example 4. Using XIncludes for shared texts without duplicating source elements

```
<… xmlns:xi="http://www.w3.org/2001/XInclude">
…
  <para>The latest version of <application><xi:include href="shared-texts.xml" xpointer=▶
"xpath(id('product-name')/text())"/></application>
    from <xi:include href="shared-texts.xml" xpointer="xpath(id('corp-name')/text())"/>
    is <xi:include href="shared-texts.xml" xpointer="xpath(id('product-version')/text())"/>.</▶
para>
…
```

UC-2: Shared strings in attribute values

This use-cases is similar to UC-1. The only difference now is that shared text can appear in an attribute value. This is not very common requirement for DocBook document, but because of differences between element and attribute values in XML separate use-cases is created.

Current solutions

Entities

This use-case can be solved only by using internal entities[3]. This is well supported in processing tools and to some extent in authoring tools.

Example 5. Using entities for shared strings in attribute values

```
<!ENTITY product-version "3.14">
<!ENTITY product-name "FooWiz">
…
<section xreflabel="Installation of &product-name; &product-version;">
…
```

UC-3: Conditional shared text

Some documents use conditional content. Shared text then can be also subject to conditional processing.

Current solutions

Conditional sections

This use-case can be solved by using conditional sections[4] which can enclose entity definitions.
Do we need example here? It is rarely used feature.

XInclude and profiling

As typical XInclude scenarios include complete elements not just their content, it is possible to define several parallel elements with different profiling attributes set, enclose them with another element and then do inclusion.

[3] http://www.w3.org/TR/REC-xml/#sec-internal-ent
[4] http://www.w3.org/TR/REC-xml/#sec-condition-sect

Example 6. Definition file for further XInclude referencing with conditional text

```
<article xmlns="http://docbook.org/ns/docbook" version="5.0">
  <title>Shared texts</title>
  …
  <para><phrase xml:id="product-name"><phrase os="win">Windows
        Protector</phrase><phrase os="linux">Linux
        Protector</phrase></phrase></para>
  …
</article>
```

UC-4: Modularized documents

For large documents it is often impractical to edit them as a single large XML file. It is practical to split large document (e.g. book) into few smaller ones (e.g. chapters) and just include them into “master” document. This approach is also useful for reusing of boilerplate texts like legalnotices.

Current solutions

Entities

This use-case can be solved by using external entities[5]. This is well supported in processing tools and to some extent in authoring tools. Also entities and their references are not part of XDM so they do not survive XSLT processing which makes various “document massaging” task challenging. External entities doesn't work very well if there are multiple levels of inclusions as entities can be declared only in the “main” document.

Example 7. Using entities for modularized documents

```
…
<!ENTITY chapter1 SYSTEM "chapter1.xml">
<!ENTITY chapter2 SYSTEM "chapter2.xml">
…
<book …>
  <info>
    <title>A Book</title>
  </info>

  &chapter1;
  &chapter2;

  <index/>
</book>
```

XInclude

XInclude[6] works very well for this use-case.

[5] http://www.w3.org/TR/REC-xml/#sec-external-ent
[6] http://www.w3.org/TR/xinclude/

Example 8. Using XIncludes for modularized documents

```
<book … xmlns:xi="http://www.w3.org/2001/XInclude">
  <info>
    <title>A Book</title>
  </info>

  <xi:include href="chapter1.xml"/>
  <xi:include href="chapter2.xml"/>

  <index/>
</book>
```

UC-5: Repeated transclusion in one “master” document

Some types of documentation are highly modular and composed from large number of small units. These units can be whole chapters, smaller self-standing units like section or topic, or it can be quite small chunk of content like admonition. If each such unit is used only once in document then there is no problem and we are working only with highly modular document (see UC-4). But if one unit is included more then once then we can face several problems.

If elements in units have assigned unique IDs (by using `xml:id` attribute) then resulting document after transclusion contains duplicate IDs. This makes file technically invalid and processing tools have difficulties in interpreting cross-references found in document.

There are different strategies which can be used to managed units and IDs which are transcluded more then once:

- Do nothing. This of course doesn't solve problem and result of processing is unpredictable.
- Preserve duplicated IDs only on their first occurrence (in document order). Other then first occurrences of ID are deleted. All links will then point to the first location in final composed document.
- Make each ID unique within transcluded unit (for example by prepending unit specific prefix to each ID). In this cases targets of all cross-references have to be adjusted. Several adjustment strategies are possible – link to the first occurrence, link to the “closest occurrence” (e.g. find minimum subtree rooted at ancestors that contains link target).

Current solution

Unfortunately there is no satisfying solution to this use-case. DocBook XSL stylesheet can be customized to link only to the first occurrence of ID (see http://www.sagehill.net/docbookxsl/DuplicateIDs.html).

UC-6: Transclusion of foreign content

Sometimes it is necessary to include foreign content into DocBook document. Examples of such foreign content are:

- listing of program source;
- content in different vocabulary like DITA or TEI which has to be transformed into DocBook prior transclusion.

Current solution

Text files with programlistings can be transcluded by DocBook `textdata` element or using `parse="text"` functionality of XInclude. There is no standardized way of transclusion of different vocabularies. Currently this has to be solved on application level.

Evaluation of current technologies

It is evident that the existing technologies are not able to handle use-cases UC-5 and UC-6. All other use-cases except UC-2 can be technically solved using XInclude. However XInclude usage is quite cumbersome. It seems that DocBook specific transclusion mechanism is needed. Proposal of such mechanism is now discussed by DocBook TC and it is available for wider review and comments at http://docbook.org/docs/transclusion/.