DocBook Transclusion

20 April 2011

This version:
http://docbook.org/docs/transclusion/2011-04-20/
Latest version:
http://docbook.org/docs/transclusion/
Previous version:
http://docbook.org/docs/transclusion/2010-12-09/
Author:
Jirka Kosek, <jirka@kosek.cz>

Table of Contents

This document describes syntax, semantics and processing model of DocBook transclusion mechanism. Please be aware that this is early stage draft – everything described below might change or disappear completely. This proposal tries to resolve *Requirements for transclusion in DocBook*[1]. DocBook TC welcomes any feedback on this draft, especially from users and developers of DocBook authoring and processing tools. Plese direct your comments to DocBook mailing list by sending email to <docbook@lists.oasis-open.org>.
Actually, for now the document is written more like tutorial. If DocBook TC decides to incorporate this into DocBook, more formal and precise specification will follow.

Transclusion in documents is described by `ref` element which references the content to transclude. There are two basic types of reference – inline and external. Inline references reference content which is defined in some other place using the `definitions` element. An external reference references some external content which might or might not be written using DocBook vocabulary.

Inline reference

An inline reference is denoted by the `ref` element with a mandatory `name` attribute which contains the name of referenced content. This name is case-sensitive and the document must contain a definition of the referenced content in the `definitions` element.

[1] http://docbook.org/docs/transclusion-requirements/

Example 1. Simple usage of inline references and defintions

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <definitions>
      <def name="product-version">3.14</def>
      <def name="product-name">FooWiz</def>
      <def name="corp-name">ACME Inc.</def>
    </definitions>
  </info>
  <para>The latest version of <application><ref name="product-name"/></application>
    from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
</article>
```

The result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
  </info>
  <para>The latest version of <application>FooWiz</application> from ACME Inc. is 3.14.</para>
</article>
```

The definition can contain arbitrary markup, not just text. All child nodes of the `def` element are transcluded by corresponding `ref` element.

Example 2. An inline definition with nested markup

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <author><ref name="jirka"/></author>
    <definitions>
      <def name="jirka">
        <personname><firstname>Jirka</firstname><surname>Kosek</surname></personname>
      </def>
    </definitions>
  </info>
  <para>This article was written by <ref name="jirka"/> when there was 25C outside.</para>
</article>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <author>
      <personname><firstname>Jirka</firstname><surname>Kosek</surname></personname>
    </author>
  </info>
  <para>This article was written by
        <personname><firstname>Jirka</firstname><surname>Kosek</surname></personname>
       when there was 25C outside.</para>
</article>
```

Definitions can be placed directly inside an `info` element or they can be stored in a separate file which can be referenced by multiple documents.

Example 3. Definitions stored in a separate document (`definitions.001.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://docbook.org/ns/docbook">
  <def name="product-version">3.14</def>
  <def name="product-name">FooWiz</def>
  <def name="corp-name">ACME Inc.</def>
</definitions>
```

Example 4. Usage of definitions stored in a separate document

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <!-- Definitions are loaded from external file -->
    <definitions definitionfile="definitions.001.xml"/>
  </info>
  <para>The latest version of <application><ref name="product-name"/></application>
    from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
</article>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
  </info>
  <para>The latest version of <application>FooWiz</application>
    from ACME Inc. is 3.14.</para>
</article>
```

Definitions from external file can be locally redefined.

Example 5. Redefinition of definition provided in an external file

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <definitions definitionfile="definitions.001.xml">
      <!-- Local definition will override definition from external definitions file -->
      <def name="product-version">4.01</def>
    </definitions>
  </info>
  <para>The latest version of <application><ref name="product-name"/></application>
    from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
</article>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
  </info>
  <para>The latest version of <application>FooWiz</application>
    from ACME Inc. is 4.01.</para>
</article>
```

Definitions can be conditional. All effectivity attributes can be used to set conditions on definitions.
Gershon: I'd like to see examples 14 and 15 be followed by a more complex exmaple that demonstrates the different result depending on processing order, so that we can also determine which order should be normative and correct per the DocBook spec.

Example 6. Conditional definitions

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <definitions>
      <def name="product-version">3.14</def>
      <!-- Conditional definitions -->
      <def name="product-name" os="win">Windows Protector</def>
      <def name="product-name" os="linux">Linux Protector</def>
      <def name="corp-name">ACME Inc.</def>
    </definitions>
  </info>
  <para>The latest version of <application><ref name="product-name"/></application>
    from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
</article>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
  </info>
  <para>The latest version of <application>Linux Protector</application>
    from ACME Inc. is 3.14.</para>
</article>
```

Effectivity attributes can be also used on references itself.

Example 7. Conditional references

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
    <definitions>
      <def name="product-version">3.14</def>
      <def name="product-name-win">Windows Protector</def>
      <def name="product-name-lin">Linux Protector</def>
      <def name="corp-name">ACME Inc.</def>
    </definitions>
  </info>
  <para>The latest version of
    <!-- Conditional references -->
   <application><ref os="win" name="product-name-win"/><ref os="linux" name="product-name-lin"/▶
></application>
    from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
</article>
```

Definitions are valid only in the subtree rooted at the parent of `info` element containing definitions.

Example 8. Scope of defintions

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <info>
    <title><ref name="corp-name"/> company product guide</title>
    <!-- Multiple definitions in single document -->
    <definitions>
      <def name="corp-name">ACME Inc.</def>
    </definitions>
  </info>
  <article>
    <info>
      <title><ref name="product-name"/> Guide</title>
      <definitions>
        <def name="product-version">3.14</def>
        <def name="product-name">FooWiz</def>
      </definitions>
    </info>
    <para>The latest version of <application><ref name="product-name"/></application>
      from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
  </article>
  <article>
    <info>
      <title><ref name="product-name"/> Guide</title>
      <definitions>
        <def name="product-version">4.2</def>
        <def name="product-name">Knit-o-Matic</def>
      </definitions>
    </info>
    <para>The latest version of <application><ref name="product-name"/></application>
      from <ref name="corp-name"/> is <ref name="product-version"/>.</para>
  </article>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>ACME Inc. company product guide</title>
  </info>
  <article>
    <info>
      <title>FooWiz Guide</title>
    </info>
    <para>The latest version of <application>FooWiz</application>
      from ACME Inc. is 3.14.</para>
  </article>
  <article>
    <info>
```

```
      <title>Knit-o-Matic Guide</title>
    </info>
    <para>The latest version of <application>Knit-o-Matic</application>
      from ACME Inc. is 4.2.</para>
  </article>
</book>
```

A reference can point to particular external definitions file not to all in-scope definitions.

Example 9. Reference to particular definition file

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
  </info>
  <para>This is article about history of <ref definitionfile="definitions.001.xml" name=▶
"corp-name"/>.</para>
</article>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <info>
    <title>Transclusions demo</title>
  </info>
  <para>This is article about history of ACME Inc..</para>
</article>
```

Procedure 1. Finding the reference definition for `ref` element with just `name` attribute

1. A transclusion processor implementation might provide a mechanism for overriding any definition. This mechanism is used first when finding a definition for a reference.
2. The closest element containing `info/definitions` is found on the ancestor-or-self:: XPath axis.
    a. If `definitions` contains `definitionfile` attribute then content of referenced file is treated as if it was inserted before the respective `definitions` element.
    b. If there are multiple matching `info/definitions/def` elements then the last one is used. If there is no matching definition then we continue with Step 2.
3. If no matching definition was find so far, an error is raised.

Procedure 2. Finding reference definition for `ref` element with `definitionfile` attribute

1. Referenced definition file is searched for definition (`def` element with matching value in a `name` attribute). *Should be definitionfile attribute supported here for chaining of definitions? Probably yes.*
2. If no matching definition is found, error is raised.

External references

An external reference is denoted by the `ref` element with a mandatory `fileref` attribute which contains URI of referenced document. The reference is replaced with content of referenced document. The outermost transcluded element is augmented with `xml:base` attribute in order to retain relative references.

Example 10. External reference

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook" xml:id="art-1">
  <title>Sample article</title>
  <para>Leading text</para>
  <ref fileref="module.001.xml"/>
</article>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<article xmlns="http://docbook.org/ns/docbook">
  <title>Sample article</title>
  <para>Leading text</para>
  <section xml:base="file:/e:/texts/docbook/tc/transclusions/module.001.xml">
    <title>Module which is going to be shared across several documents</title>
    <para>Bla bla bla</para>
  </section>
</article>
```

It is possible to specify additional attributes `xpointer`, `parse` and `encoding` – their meaning is identical to corresponding attributes from XInclude[2].

Please note that while process similar to XInclude Base URI Fixup[3] is performed, Language Fixup[4] is not performed.[5]

[2] http://www.w3.org/TR/xinclude/
[3] http://www.w3.org/TR/xinclude/#base
[4] http://www.w3.org/TR/xinclude/#language
[5] This effectively means that it is not necessary to specify `xml:lang` on each module.

Example 11. Transclusion of document which contains further references

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Sample book</title>
  <ref fileref="article.001.xml"/>
  <ref fileref="shared-texts.002.xml"/>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Sample book</title>
  <article xmlns:ta="http://docbook.org/xslt/ns/transclusion-annotation"
    xml:base="file:/e:/texts/docbook/tc/transclusions/article.001.xml">
    <title>Sample article</title>
    <para>Leading text</para>
    <section xml:base="file:/e:/texts/docbook/tc/transclusions/module.001.xml">
      <title>Module which is going to be shared across several documents</title>
      <para>Bla bla bla</para>
    </section>
  </article>
  <article xmlns:ta="http://docbook.org/xslt/ns/transclusion-annotation"
    xml:base="file:/e:/texts/docbook/tc/transclusions/shared-texts.002.xml">
    <info>
      <title>Transclusions demo</title>
      <author>
        <personname><firstname>Jirka</firstname><surname>Kosek</surname></personname>
      </author>
    </info>
    <para>This article was written by
        <personname><firstname>Jirka</firstname><surname>Kosek</surname></personname> when there
      was 25C outside.</para>
  </article>
</book>
```

Special ID/IDREF processing

Transcluded content can contain an `xml:id` attributes. If one fragment is transcluded more then once then the resulting document after transclusion will contain duplicate IDs. The same problem may arise if IDs are colliding in transcluded modules. To overcome this problem all IDs and references to them can be adjusted during the transclusion process.

If there is `xml:id` attribute present on the `ref` element, then this ID will replace `xml:id` on the outermost element of any transcluded content. It is an error to transclude content which is not enclosed in a single element and specifying an `xml:id` attribute on `ref` element at the same time.

How IDs are going to be adjusted during transclusion is controlled by the `idfixup` attribute on the `ref` element. It can have one of the following values.

`none` No ID adjustment is done

`strip` All IDs are stripped (except `xml:id` inherited from `ref` element)

`prefix` All IDs are prefixed with a value specified in `prefix` attribute

`auto` All IDs are prefixed with a value which is unique for each `ref` element[6]

This is default behavior if attribute is not specified.

Of course if IDs are adjusted then all corresponding references has to be also corrected. This is controlled by `linkscope` attribute. It can have one of the following values.

`user` No IDREF adjustment is done

`local` All IDREFs in transcluded content are prefixed by user specified prefix (when `idfixup="prefix"`) or auto-generated prefix (when `idfixup="auto"`).

Using this value with other `idfixup` values is an error.*Maybe raising error is to strict approach.*

`near` All IDREFs in transcluded content are adjusted to point to the closest element which has a matching ID. A matching ID doesn't mean string equality between ID and IDREF values – it is sufficient if second part of ID and IDREF after removal of possibly added prefixes is matching.

When searching for the closest element ancestor elements of an element with an IDREF attribute are gradually inspected and matching ID is searched between all their descendants. If there is no matching ID, then parent is inspected and so on until match is found or document root is reached.

`global` All IDREFs in transcluded content are adjusted to point to the first element in document order which has a matching ID. A matching ID doesn't mean string equality between ID and IDREF values – it is sufficient if the second part of ID and IDREF after removal of possibly added prefixes are matching.

By using various combinations of `idfixup` and `linkscope` attributes we can achieve different linking behavior. The following examples show the effect of using those two attributes. Examples are transcluding the following procedure which contains one internal link and one external (points to target outside of this module).

Example 12. Module with sample procedure

```
<?xml version="1.0" encoding="UTF-8"?>
<procedure xmlns="http://docbook.org/ns/docbook" xml:id="paper-insert">
  <title>Inserting paper into printer</title>
  <para>This procedure is targeted to printer owners.
    If you don't have printer, consider <link linkend="buy">buying one</link>.</para>
  <step xml:id="s1"><para>Make sure that you have paper.</para></step>
  <step><para>Insert paper into printer. If you don't have paper consult <xref linkend="s1"/></▶
para></step>
</procedure>
```

Now lets assume that we want to transclude this module twice to show how we can deal with duplicate IDs problem.

[6]For example XSLT based implementations can use `generate-id()` on `ref` element to generate such unique prefix.

Example 13. Automatic ID/IDREF adjustment

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="d2e22---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="d2e22---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e22---s1"
         /></para>
      </step>
    </procedure>
  </chapter>
  <chapter>
```

```
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure xml:id="d2e36---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="d2e36---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e36---s1"
          /></para>
      </step>
    </procedure>
  </chapter>
</book>
```

We haven't specified `idfixup` and `linkscope` attributes so the default behavior is triggered. All IDs in transcluded modules are automatically prefixed to prevent ID collisions. Then IDREFs are fixed so that links point to the nearest possible target. For example the link from step 2 to step 1 in procedure always points to the same instance of procedure. However “buy” link is pointing correctly to target in the main document.

Example 14. Global linkscope

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml" linkscope="global"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="d2e22---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="d2e22---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e22---s1"
          /></para>
      </step>
    </procedure>
  </chapter>
  <chapter>
```

```
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure xml:id="d2e36---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="d2e36---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e22---s1"
          /></para>
      </step>
    </procedure>
  </chapter>
</book>
```

We used `linkscope="global"` on the second transclusion. Result is that link from step 2 in the second procedure now links to step 1 in the first procedure.

Example 15. Local linkscope

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml" linkscope="local"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="d2e22---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
         linkend="d2e22---buy">buying one</link>.</para>
      <step xml:id="d2e22---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e22---s1"
         /></para>
      </step>
    </procedure>
  </chapter>
  <chapter>
```

```
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure xml:id="d2e36---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="d2e36---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e36---s1"
          /></para>
      </step>
    </procedure>
  </chapter>
</book>
```

We used `linkscope="local"` on the first transclusion. This means that no link from this transclusion can point outside of this transclusion. Because there was such link (“buy” link), thus the result of transclusion is broken because there is no corresponding target for IDREF `d2e22---buy`.

Example 16. Manually assigned prefix

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml" xml:id="install-proc" idfixup="prefix" prefix=▶
"install-proc_"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml" xml:id="maintain-proc" idfixup="prefix" prefix=▶
"maintain-proc_"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="install-proc_paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="install-proc_s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref
            linkend="install-proc_s1"/></para>
      </step>
    </procedure>
  </chapter>
```

```
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure xml:id="maintain-proc_paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="maintain-proc_s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref
            linkend="maintain-proc_s1"/></para>
      </step>
    </procedure>
  </chapter>
</book>
```

If we care about the resulting IDs after transclusion we can manually assign some meaningful prefix before IDs in transcluded document.

Example 17. Disabling ID fixup

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml" idfixup="none"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml" idfixup="none"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
         linkend="buy">buying one</link>.</para>
      <step xml:id="s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
       <para>Insert paper into printer. If you don't have paper consult <xref linkend="s1"/></▶
para>
      </step>
    </procedure>
  </chapter>
  <chapter>
    <title>Maintenance</title>
```

```
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure xml:id="paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
       <para>Insert paper into printer. If you don't have paper consult <xref linkend="s1"/></▶
para>
      </step>
    </procedure>
  </chapter>
</book>
```

We have disabled ID fixup by `idfixup="none"`. The resulting document thus contain duplicated IDs.

Example 18. Stripping IDs

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml" idfixup="strip"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="d2e22---paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step xml:id="d2e22---s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e22---s1"
          /></para>
      </step>
    </procedure>
  </chapter>
  <chapter>
```

```
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step>
        <para>Make sure that you have paper.</para>
      </step>
      <step>
        <para>Insert paper into printer. If you don't have paper consult <xref linkend=▶
"d2e22---s1"
          /></para>
      </step>
    </procedure>
  </chapter>
</book>
```

We have stripped all IDs from the second transcluded procedure by `idfixup="strip"`. Thus there are no duplicate IDs in the resulting document and links from second procedure always target first one. However IDs in the first procedure were automatically prefixed.

Example 19. Retaining original IDs

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <ref fileref="procedure.001.xml" idfixup="none" linkscope="user"/>
  </chapter>
  <chapter>
    <title>Maintenance</title>
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <ref fileref="procedure.001.xml" idfixup="strip" linkscope="user"/>
  </chapter>
</book>
```

Result of transclusion:

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://docbook.org/ns/docbook">
  <title>Definitive Printer Guide</title>
  <chapter xml:id="buy">
    <title>Buying printer</title>
    <para>Grab money, go to shop, ...</para>
  </chapter>
  <chapter>
    <title>Quick installation guide</title>
    <para>Carefully follow all procedures bellow.</para>
    <procedure xml:id="paper-insert"
      xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
         linkend="buy">buying one</link>.</para>
      <step xml:id="s1">
        <para>Make sure that you have paper.</para>
      </step>
      <step>
       <para>Insert paper into printer. If you don't have paper consult <xref linkend="s1"/></▶
para>
      </step>
    </procedure>
  </chapter>
  <chapter>
    <title>Maintenance</title>
```

```
    <para>Be friendly to your printer when you speak to it.</para>
   <para>If green led is blinking, please add missing paper using the following procedure.</para>
    <procedure xml:base="file:/e:/texts/docbook/tc/transclusions/procedure.001.xml">
      <title>Inserting paper into printer</title>
     <para>This procedure is targeted to printer owners. If you don't have printer, consider ▶
<link
          linkend="buy">buying one</link>.</para>
      <step>
        <para>Make sure that you have paper.</para>
      </step>
      <step>
       <para>Insert paper into printer. If you don't have paper consult <xref linkend="s1"/></▶
para>
      </step>
    </procedure>
  </chapter>
</book>
```

We have stripped all IDs from the second transcluded procedure by `idfixup="strip"`. Thus there are no duplicate IDs in the resulting document and links from second procedure always target first one. IDs in the first procedure were not automatically prefixed since we have specified `idfixup="none"`.

Should be there option to preserve original IDs (without prefix) in output when doing prefixes?

Transformations

FIXME: This should provide some very generic mechanism for applying various transformations on referenced content before actual transclusion. For example DITA -> DocBook, V4.x -> V5.0 conversion and so on.

FIXME: TC decided this is not in scope of transclusions. Sources should be prepared in advance.

Evaluation

The above DocBook transclusion proposal solves all use cases[7] except UC-2[8]. Transclusions are more usable and can solve various problems introduced by multiple inclusion of the same content. Some of those problems can be solved by using XInclude – but syntax is cumbersome and there is no good interoperability between various XInclude implementations.

For the above reasons I think that transclusions should be added into core DocBook. I don't think that making them separate XML transclusion standard is viable approach. A transclusion processor has to know which attributes are of ID/IDREF type for each document type. History shown that generic standards relying on access to a schema are not successful.

[7] http://docbook.org/docs/transclusion-requirements/
[8] http://docbook.org/docs/transclusion-requirements/#uc-2

A. DocBook schema with support for transclusions

```
# TODO
# - use DocBook schema coding style

default namespace db = "http://docbook.org/ns/docbook"

include "docbook.rnc"

# element for referencing content
db.ref =
  element ref {
    (db.ref.inline | db.ref.external)
  }

# common content
db.ref.common =
  db.effectivity.attributes,
  attribute xml:id { xsd:ID }?,
  attribute linkscope { "user" | "local" | "near" | "global"}?,
  attribute idfixup { "none" | "strip" | "prefix" | "auto" }?,
  attribute prefix { xsd:NCName }?,
  (db.transform*)

# inline content is grabed from local definitions or from external definitions file
db.ref.inline =
  attribute definitionfile { xsd:anyURI }?,
  attribute name { xsd:NCName },
  db.ref.common

# content from external sources
db.ref.external =
  attribute fileref { xsd:anyURI },
  attribute xpointer { text }?,
  attribute encoding { text }?,
  attribute parse { "xml" | "text" }?,
  db.ref.common

# FIXME: specify all kinds of possible transformations
db.transform = empty

# element for defitions
db.definitions =
  element definitions {
    attribute definitionfile { xsd:anyURI }?,
    db.def*
  }

# single definition
db.def =
```

```
  element def {
    attribute name { xsd:NCName },
    db.effectivity.attributes,
    (text & db._any*)
  }

# hooks into standard schema

# allow references to appear almost everywhere
db.ubiq.inlines |= db.ref*

db.divisions |= db.ref
db.components |= db.ref

db.toplevel.sections |=
   (db.section|db.ref)+ | (db.simplesect|db.ref)+

db.recursive.sections |=
   (db.section|db.ref)+ | (db.simplesect|db.ref)+

db.part.components |= db.ref

db.reference.components |= db.ref

db.all.blocks |= db.ref

db.info.elements |= db.ref

# definitions can be only inside info elements
db.info.elements |= db.definitions

# or they can be provided in a separate file
start |= db.definitions
```

Should we allow multiple file references in @definitionfile?
Should we allow ref inside def? (Probably yes).

B. Sample transclusion processor written in XSLT 2.0

Please note that this sample transclusion processor is not yet feature complete. It supports only subset of proposal.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
  xmlns:f="http://docbook.org/xslt/ns/extension"
  xmlns:mp="http://docbook.org/xslt/ns/mode/private"
  xmlns:db="http://docbook.org/ns/docbook"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:ta="http://docbook.org/xslt/ns/transclusion-annotation"
  exclude-result-prefixes="f mp xs">

<xsl:template match="/">

  <xsl:variable name="expanded" select="f:expand-definitions(/)"/>

  <xsl:variable name="resolved" select="f:resolve-references($expanded)"/>

  <xsl:variable name="result" select="f:adjust-idrefs($resolved)"/>

  <xsl:sequence select="$result"/>

</xsl:template>

<!-- Separator for auto generated prefixes -->
<xsl:param name="psep" select="'---'"/>

<xsl:function name="f:transclude" as="node()+">
  <xsl:param name="doc" as="node()+"/>

  <xsl:variable name="expanded" select="f:expand-definitions($doc)"/>
  <xsl:variable name="resolved" select="f:resolve-references($expanded)"/>
  <xsl:sequence select="$resolved"/>
</xsl:function>

<xsl:function name="f:expand-definitions" as="node()+">
  <xsl:param name="doc" as="node()+"/>

  <xsl:apply-templates select="$doc" mode="mp:expand-definitions"/>
</xsl:function>

<xsl:template match="node()" mode="mp:expand-definitions">
  <xsl:copy>
    <xsl:copy-of select="@*"/>
    <xsl:apply-templates mode="mp:expand-definitions"/>
  </xsl:copy>
</xsl:template>

<xsl:template match="db:definitions[@definitionfile]" mode="mp:expand-definitions">
  <xsl:copy-of select="f:expand-definitions(doc(@definitionfile))"/>
  <xsl:copy>
    <xsl:copy-of select="@* except @definitionfile"/>
    <xsl:apply-templates mode="mp:expand-definitions"/>
  </xsl:copy>
</xsl:template>

<xsl:function name="f:resolve-references" as="node()+">
  <xsl:param name="doc" as="node()+"/>

  <xsl:apply-templates select="$doc" mode="mp:transclude"/>
</xsl:function>

<xsl:template match="node()" mode="mp:transclude">
```

```
  <xsl:param name="idfixup" select="'auto'" tunnel="yes"/>
  <xsl:param name="prefix" tunnel="yes"/>
  <xsl:copy>
    <xsl:copy-of select="@* except @xml:id"/>
    <xsl:if test="@xml:id">
      <xsl:choose>
 <xsl:when test="($idfixup = 'none') or @ta:linkscope">
   <xsl:copy-of select="@xml:id"/>
 </xsl:when>
 <xsl:when test="$idfixup = 'strip'">
 </xsl:when>
 <xsl:otherwise>
   <xsl:attribute name="xml:id" select="concat($prefix, @xml:id)"/>
 </xsl:otherwise>
      </xsl:choose>
    </xsl:if>
    <xsl:apply-templates mode="mp:transclude">
     <xsl:with-param name="prefix" select="if (@ta:linkscope) then '' else $prefix"/>   <!-- ▶
Prevent adding several prefixes on multi-level inclusions -->
    </xsl:apply-templates>
  </xsl:copy>
</xsl:template>

<!-- FIMXE: this stripping is there just to have more compact output -->
<xsl:template match="db:definitions" mode="mp:transclude" priority="1">
</xsl:template>

<xsl:template match="db:ref[@name]" mode="mp:transclude">
  <xsl:variable name="name" select="@name"/>
  <xsl:variable name="content">
    <xsl:choose>
      <xsl:when test="@definitionfile">
 <xsl:variable name="defs" select="f:expand-definitions(doc(@definitionfile))"/>

 <xsl:choose>
   <xsl:when test="$defs/db:def[@name = $name]">
     <xsl:sequence select="($defs/db:def[@name = $name])[last()]/node()"/>
   </xsl:when>
   <xsl:otherwise>
     <xsl:message>Error: definition of "<xsl:value-of select="$name"/>" was not found.</▶
xsl:message>
   </xsl:otherwise>
 </xsl:choose>
      </xsl:when>
      <xsl:otherwise>
 <xsl:sequence select="f:definition-for-name(@name, .)"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <xsl:variable name="idfixup" select="if (@idfixup) then @idfixup else 'auto'"/>

  <xsl:variable name="linkscope" select="if (@linkscope) then @linkscope else 'near'"/>
```

```
  <xsl:variable name="prefix">
    <xsl:choose>
      <xsl:when test="$idfixup = 'auto'">
 <xsl:sequence select="concat(generate-id(.), $psep)"/>
      </xsl:when>
      <xsl:when test="$idfixup = 'prefix'">
 <xsl:sequence select="string(@prefix)"/>
      </xsl:when>
      <xsl:otherwise></xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <xsl:variable name="ref-xmlid" select="@xml:id"/>

  <xsl:if test="count($content/(*|text()[normalize-space(.) ne '])) > 1 and $ref-xmlid">
    <xsl:message>Error xml:id can't be added to definition without single outermost element.</▶
xsl:message>
  </xsl:if>

  <xsl:for-each select="$content/node()">
    <xsl:copy>
      <xsl:variable name="xmlid" select="if ($ref-xmlid) then $ref-xmlid else @xml:id"/>
      <xsl:copy-of select="@* except @xml:id"/>
      <xsl:if test="self::*">
 <xsl:choose>
   <xsl:when test="$idfixup = 'none' and $xmlid">
     <xsl:attribute name="xml:id" select="$xmlid"/>
   </xsl:when>
   <xsl:when test="($idfixup = 'strip' and not($ref-xmlid)) or not($xmlid)">
   </xsl:when>
   <xsl:otherwise>
     <xsl:attribute name="xml:id" select="concat($prefix, @xml:id)"/>
   </xsl:otherwise>
 </xsl:choose>
 <!-- <xsl:attribute name="xml:base" select="$baseuri"/> -->
 <xsl:attribute name="ta:linkscope" select="$linkscope"/>
 <xsl:attribute name="ta:prefix" select="$prefix"/>
      </xsl:if>
      <xsl:apply-templates mode="mp:transclude">
 <xsl:with-param name="idfixup" select="$idfixup" tunnel="yes"/>
 <xsl:with-param name="prefix" select="$prefix" tunnel="yes"/>
      </xsl:apply-templates>
    </xsl:copy>
  </xsl:for-each>
</xsl:template>

<xsl:function name="f:definition-for-name" as="node()*">
  <xsl:param name="name" as="xs:string"/>
  <xsl:param name="context" as="node()"/>

  <xsl:variable name="closest-info-with-defs" select="$context/ancestor-or-self::*/▶
db:info[db:definitions][1]"/>

  <xsl:choose>
```

```
    <xsl:when test="$closest-info-with-defs">
      <xsl:choose>
 <xsl:when test="$closest-info-with-defs/db:definitions/db:def[@name = $name]">
  <xsl:sequence select="($closest-info-with-defs/db:definitions/db:def[@name = $name])[last()]/▶
node()"/>
 </xsl:when>
 <xsl:otherwise>
   <xsl:sequence select="f:definition-for-name($name, $closest-info-with-defs/../..)"/>
 </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>Error: definition of "<xsl:value-of select="$name"/>" was not found.</▶
xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>

<xsl:template match="db:ref[@parse eq 'text']" mode="mp:transclude">
  <xsl:choose>
    <xsl:when test="@encoding">
      <xsl:sequence select="unparsed-text(@fileref, @encoding)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:sequence select="unparsed-text(@fileref)"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>


<xsl:template match="db:ref[@fileref and (@parse eq 'xml' or not(@parse))]" mode="mp:transclude">
  <xsl:variable name="doc">
    <xsl:choose>
      <xsl:when test="@xpointer">
 <xsl:sequence select="f:transclude(doc(@fileref)//*[@xml:id = current()/@xpointer])"/>
      </xsl:when>
      <xsl:otherwise>
 <xsl:sequence select="f:transclude(doc(@fileref))"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <xsl:variable name="baseuri" select="resolve-uri(@fileref, base-uri(.))"/>

  <xsl:variable name="idfixup" select="if (@idfixup) then @idfixup else 'auto'"/>

  <xsl:variable name="linkscope" select="if (@linkscope) then @linkscope else 'near'"/>

  <xsl:variable name="prefix">
    <xsl:choose>
      <xsl:when test="$idfixup = 'auto'">
 <xsl:sequence select="concat(generate-id(.), $psep)"/>
      </xsl:when>
      <xsl:when test="$idfixup = 'prefix'">
```

```
 <xsl:sequence select="string(@prefix)"/>
      </xsl:when>
      <xsl:otherwise></xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <xsl:variable name="ref-xmlid" select="@xml:id"/>

  <xsl:for-each select="$doc/node()">
    <xsl:copy>
      <xsl:variable name="xmlid" select="if ($ref-xmlid) then $ref-xmlid else @xml:id"/>
      <xsl:copy-of select="@* except @xml:id"/>
      <xsl:if test="self::*">
 <xsl:choose>
   <xsl:when test="$idfixup = 'none' and $xmlid">
     <xsl:attribute name="xml:id" select="$xmlid"/>
   </xsl:when>
   <xsl:when test="($idfixup = 'strip' and not($ref-xmlid)) or not($xmlid)">
   </xsl:when>
   <xsl:otherwise>
     <xsl:attribute name="xml:id" select="concat($prefix, @xml:id)"/>
   </xsl:otherwise>
 </xsl:choose>
 <xsl:attribute name="xml:base" select="$baseuri"/>
 <xsl:attribute name="ta:linkscope" select="$linkscope"/>
 <xsl:attribute name="ta:prefix" select="$prefix"/>
      </xsl:if>
      <xsl:apply-templates mode="mp:transclude">
 <xsl:with-param name="idfixup" select="$idfixup" tunnel="yes"/>
 <xsl:with-param name="prefix" select="$prefix" tunnel="yes"/>
      </xsl:apply-templates>
    </xsl:copy>
    <xsl:copy-of select="following-sibling::node()"/>
  </xsl:for-each>
</xsl:template>

<xsl:function name="f:adjust-idrefs" as="node()+">
  <xsl:param name="doc" as="node()+"/>

  <xsl:apply-templates select="$doc" mode="mp:adjust-idrefs"/>
</xsl:function>

<xsl:template match="node()|@*" mode="mp:adjust-idrefs">
  <xsl:copy>
    <xsl:apply-templates select="@* | node()" mode="mp:adjust-idrefs"/>
  </xsl:copy>
</xsl:template>

<!-- FIXME: add support for @linkends, @zone, @areарefs -->
<!-- FIEMX: add support for xlink:href starting with # -->
<xsl:template match="@linkend | @endterm | @otherterm | @startref" mode="mp:adjust-idrefs">
  <xsl:variable name="idref" select="."/>

  <xsl:variable name="annotation" select="ancestor-or-self::*[@ta:linkscope][1]"/>
```

```
  <xsl:variable name="linkscope" select="($annotation/@ta:linkscope, 'near')[1]"/>
  <xsl:variable name="prefix" select="$annotation/@ta:prefix"/>

  <xsl:attribute name="{local-name(.)}">
    <xsl:choose>
      <xsl:when test="$linkscope = 'user'">
 <xsl:value-of select="$idref"/>
      </xsl:when>
      <xsl:when test="$linkscope = 'local'">
 <xsl:value-of select="concat($prefix, $idref)"/>
      </xsl:when>
      <xsl:when test="$linkscope = 'near'">
 <xsl:value-of select="f:nearest-matching-id($idref, ..)"/>
      </xsl:when>
      <xsl:when test="$linkscope = 'global'">
 <xsl:value-of select="f:nearest-matching-id($idref, root(.))"/>
      </xsl:when>
    </xsl:choose>
  </xsl:attribute>
</xsl:template>

<xsl:function name="f:nearest-matching-id" as="xs:string?">
  <xsl:param name="idref" as="xs:string"/>
  <xsl:param name="context" as="node()"/>

  <!-- FIXME: key() requires document-node() rooted subtree -->
  <!--  <xsl:variable name="targets" select="key('unprefixed-id', f:unprefixed-id($idref, ▶
$context), $context)"/> -->
  <xsl:variable name="targets" select="$context//*[@xml:id][f:unprefixed-id(@xml:id, .) eq ▶
f:unprefixed-id($idref, $context)]"/>

  <xsl:choose>
    <xsl:when test="not($targets) and $context/..">
      <xsl:sequence select="f:nearest-matching-id($idref, $context/..)"/>
    </xsl:when>
    <xsl:when test="$targets">
      <xsl:sequence select="$targets[1]/string(@xml:id)"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:message>Error: no matching ID for reference "<xsl:value-of select="$idref"/>" was ▶
found.</xsl:message>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>

<!-- FIXME: type annotation should be without ?, find why it is called with empty sequence -->
<xsl:function name="f:unprefixed-id" as="xs:string?">
  <xsl:param name="id" as="xs:string?"/>
  <xsl:param name="context" as="node()"/>

  <xsl:variable name="prefix" select="$context/ancestor-or-self::*[@ta:prefix][1]/@ta:prefix"/>

  <xsl:sequence select="if ($prefix) then substring-after($id, $prefix) else $id"/>
</xsl:function>
```

```
<!--
<xsl:key name="unprefixed-id" match="*[@xml:id]" use="f:unprefixed-id(@xml:id, .)"/>
-->

</xsl:stylesheet>
```

C. Mapping Transclusions to XInclude

Some basic functionality of transclusions can be directly mapped into XInclude as show in the table below. However there are still some very important features which can't be replicated with pure XInclude, namely:

- ID/IDREF fixup – it was one of the most important requirements to provide solution to the duplicate IDs problem;
- profiling – it's not possible to specify profiling attributes on `xi:include` element as it is subject to special handling;
- changing ID during transclusion – it's not possible to specify new xml:id value for transcluded content on `xi:include` element as it is subject to special handling;
- redefinitions are not supported. Having more definitions in one file will lead to an invalid file as each definition has to be identified by an unique `xml:id`.

Table C.1. Mapping Transclusions to XInclude

Transclusion construct	XInclude equivalent	Note
`<ref name="foo"/>`	`<xi:include xpointer="xpath(id(foo)/node())"/>`	XInclude implementations are not required to support missing `href` attribute. See http://www.w3.org/TR/xinclude/#include-location
`<ref definitionfile="bar" name="foo"/>`	`<xi:include href="bar" xpointer="xpath(id(foo)/node())"/>`	
`<ref fileref="foo"/>`	`<xi:include href="foo"/>`	
`<definitions definitionfile="foo"/>`	`<xi:include href="foo"/>`	DocBook schema and stylesheets has to be modified to ignore elements which are inserted only for further referencing.
`<def name="foo">…</def>`	`<phrase xml:id="foo">…</phrase>`	Any generic DocBook element can be used instead of `phrase`. Element itself is not transcluded.

Note

Please note that `xpath()` XPointer scheme is not widely supported. Formerly `xpointer()` schema was popular although it's standardization was never finished. Later on `xpath()` schema appeared in the list of registered schemes http://www.w3.org/2005/04/xpointer-schemes/. Today two special schemes exists – `xpath1()` and `xpath2()` – for respective versions of XPath language.

D. Alternative proposal from Hussein Shafie

This appendix summarizes points made in the following email http://lists.oasis-open.org/archives/docbook/201012/msg00014.html.

Any element may bear one of these two attributes: `ref` and `copy`. The value of these attributes is an URL, possibly ending with a fragment.

When an element has a `ref` or a `copy` attribute, it must be completely empty.

“Compose” transclusion directive

`ref` is a “compose” transclusion directive. Examples:

```
<chapter ref="chapter1.xml"/>

<section ref="book.xml#api_reference"/>
```

When `ref` attribute is used, there is no ID/IDREF fixup in the transcluded content.

“Instantiate a copy” transclusion directive

`copy` is an “instantiate a copy” transclusion directive. Examples:

```
<note xml:id="warning" copy="common.xml#legal_warning"/>

<phrase copy="common.xml#product_name"/>
```

When `copy` attribute is used, there is an automatic ID/IDREF fixup in the transcluded content, the one described above:

- Add a globally unique suffix to each ID defined in the copy.
- Update the IDREF/IDREFS which point to these IDs.
- Ignore the IDREF/IDREFS which point outside the copy.

Evaluation

Although very simple, this proposal covers many use-cases. Although compared to the original proposal several features are missing:

- It is not possible to include just a text node, or sequence of sibling nodes. Only one element (and its content) can be transcluded at one time.
- It is not possible to override definitions.
- It is not possible to configure way in which ID fixup is done.
- Verbosity for including simple inline content is similar to XInclude.