

Casting a critical eye on the Next Big Thing in technical publishing.

Building Efficient Multilingual Workflows

BY SARAH O'KEEFE, Associate Fellow

In building a business case for XML-based publishing, one common argument is that localization will be more efficient because document production tasks are streamlined and automated. This column describes two technology standards you might use in multilingual workflows: the Extensible Stylesheet Language (XSL) and XML Localization Interchange File Format (XLIFF).



Formatting Automation with XSL

If you have content in XML format, you can use XSL to transform the XML into deliverable formats, such as HTML. For example, given the following XML content:

```
<note>This is a note</note>
```

You can transform the XML content into the HTML markup shown here:

```
<p><b>NOTE:</b> This is a note</p>
```

To do so, you need a simple transformation template, shown here:

```
<xsl:template match="note">
  <b>NOTE:</b>
  <xsl:apply-templates/>
</xsl:template>
```

The XSL code in the preceding example, however, would only work in English. For other languages, you need to modify the “NOTE:” text. Instead of embedding the note text directly into the template, you can set up the template to use a variable, as shown here:

```
<xsl:template match="note">
  <b><xsl:value-of select="$notetext"/></b>
  <xsl:apply-templates/>
</xsl:template>
```

Then you can define the variable value based on the language of your document (this example assumes that you have already specified the language in the “\$language” variable):

```
<xsl:variable name="notetext">
  <xsl:choose>
    <xsl:when test="$language = 'en'">NOTE:</xsl:when>
    <xsl:when test="$language = 'de'">HINWEIS:</xsl:when>
    ...
  </xsl:choose>
</xsl:variable>
```

The “en” setting is for English; the “de” setting is for German (Deutsch).

If your template needs to support more than a few languages, you’ll end up with a lengthy <xsl:choose> statement for each repeated text item, such as note, caution, warning, table of contents, index, next, and previous. Instead, consider creating a file that contains all of the text strings. For the note and caution text, you might see the following:

```
<strings>
  <label name="note" language="en">NOTE:</label>
  <label name="note" language="de">HINWEIS:</label>
  <label name="caution" language="en">CAUTION:</label>
  <label name="caution" language="de">VORSICHT:</label>
  ...
</strings>
```

Assuming that the strings are stored in a file named labels.xml, you could define your note text variable as shown in the following example:

```
<xsl:variable name="note" select="document('labels.xml')/strings/label[@name = 'note'][@language = $language]"/>
```

This approach is much more elegant than a lengthy <xsl:choose> statement. It’s also easier to maintain; when you add a new language; you simply add a new <label> entry for each string with the appropriate name, language setting, and text value.

I recommend separating all language-specific information in your transformation templates in this way so that you can easily modify the language strings.

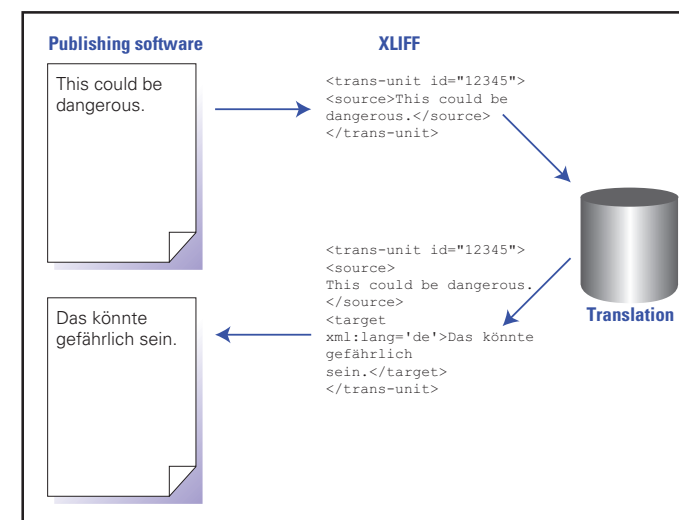
A multilingual XSL transformation process lets you convert XML source content from any supported language into final output (such as HTML) in seconds. The cost savings are potentially enormous. Consider an (hypothetical) organization that localizes 400 pages of new content every year into eight languages. (For our simplistic example, we’ll ignore the effect of translation memory.) Four hundred pages would contain approximately 250 words per page, and at \$0.25 per word per language, the total localization cost for these 400 pages would be approximately \$200,000 for the eight languages. According to Nick Rosenthal, managing director of Salford Translations Ltd. and a past president of STC’s UK Chapter, desktop publishing (that is, formatting) typically accounts for 30–50 percent of the total project cost in localization. Thus, the DTP cost component would be between \$60,000 and \$100,000. By implementing automated publishing from XML through XSL to your final output(s), you could cut that cost “all the way down to zero,” says Rosenthal. This simple example demonstrates that automating formatting could save significant time and money. Of course, you must balance the potential cost savings against the cost of implementing an XML-based workflow.

Removing Authoring Application Dependencies with XLIFF

XML Localization Interchange File Format (XLIFF) is a standard for storing and exchanging localized content. XLIFF itself is an XML vocabulary, but it is not intended for authoring or for direct use by localizers. Unlike XSL, you can use XLIFF in a non-XML-based workflow.

Localization companies work with a huge variety of proprietary formats, such as Word, InDesign, FrameMaker, QuarkXPress, and many more. For translation purposes, they generally move information into specialized translation workbench software, do the language work, and then move the translated text back into the publishing application. XLIFF offers a standard format in which to accomplish these exchanges.

Figure 1: Translation workflow with XLIFF



According to Maxwell Hoffmann, director of Document Globalization Practice at Globalization Partners International, XLIFF “parallels the intent of XML” by making localizable information portable. It does not, however, require you to implement an XML-based authoring workflow.

XLIFF files store information in chunks (or segments) using the <trans-unit> element. Most often, these chunks are paragraphs. To use XLIFF in a translation process, you first extract source content from its original format into an XLIFF file. For example, you might have the following information (with simplified markup):

```
<topic>
  <title>Ants</title>
  <p>Ants are interesting</p>
</topic>
```

You extract the various segments into an XLIFF file and put unique identifiers in as placeholders:

```
<topic>
  <title (id = "1")</title>
  <p (id = "2")</p>
</topic>
```

The XLIFF file initially contains just the source text:

```
<xliff>
  <file original="ants.xml" source-language="en" target-language="de">
  <body>
    <trans-unit id="1">
      <source>Ants</source>
    </trans-unit>
    <trans-unit id="2">
      <source>Ants are interesting</source>
    </trans-unit>
  </body>
</file>
</xliff>
```

After processing, the translation for each segment is added to the file:

```
<xliff>
  <file original="ants.xml" source-language="en" target-language="de">
  <body>
    <trans-unit id="1">
      <source>Ants</source>
      <target>Ameisen</target>
    </trans-unit>
    <trans-unit id="2">
      <source>Ants are interesting</source>
```

```

        <target>Ameisen sind interessant.</target>
    </trans-unit>
</body>
</file>
</xliff>

```

You could include multiple targets (with appropriate language attributes) in a single file or even multiple versions of a single language (such as a draft and a final version). Rosenthal notes that you can also use XLIFF to package a large collection of DITA topics and conref files into a single large file for translation management purposes. After translation, the localized content is delivered back to the various component files.

Even if you are creating content in a “traditional,” non-XML workflow, you may find that XML has insinuated itself as a content exchange format. In addition to XLIFF, the TMX standard is used for exchanging translation memory content between translation software programs. Typically, authors and localizers would not even be aware that these technologies are being used. XSL would normally be used in a workflow where source content is stored in XML, so authors would be working in a structured authoring environment. Because translators generally modify text within existing content structures, they usually do not work in XML editors. Instead, they have specialized translation software, which manages the XML tagging for them. XML’s importance in multilingual workflows is not limited to content created in XML and to structured authoring, it also facilitates the transfer of content from application to application. [i](#)

DOCUMENTATION FOR STANDARDS

XLIFF: www.oasis-open.org/committees/tc_home.php?wg_abbrev=xliff

XSL: www.w3.org/TR/xsl/

Many thanks to Maxwell Hoffmann, director of Document Globalization Practices at Globalization Partners International, and Nick Rosenthal, managing director of Salford Translations Ltd., for their assistance with this article.

Sarah O’Keefe (xmlstrategist@scriptorium.com) is founder and president of Scriptorium Publishing Services Inc. (www.scriptorium.com) based in Research Triangle Park, North Carolina. The company is focused on implementing tools and processes to optimize publishing workflows. Services include developing and deploying XML-based structured authoring environments, configuring authoring and publishing tools, and providing technical training. Her publications include Publishing Fundamentals: FrameMaker 7, The WebWorks Publisher Cookbook, Technical Writing 101, FrameMaker for Dummies, and numerous white papers.