

## **FOCUS ON STANDARDS**

In this two-part series, Andrzej Zydron outlines pitfalls that are often encountered by authors, programmers and localizers when first using XML, as well as ways to avoid these problems. Following Zydron's advice can save developers time, money and headaches, and can help them reach out effectively to the world.

---

### **Introduction**



*Andrzej Zydron*

The adoption of XML as a standard for the storage, retrieval and delivery of information has meant that many enterprises have large corpora in this format. Very often, information components in these corpora require translation. Normally, such enterprises have enjoyed all of the benefits of XML on the information creation side, but very often have failed to maximize all the benefits that XML-based translation can provide.

The separation of form and content, which is inherent within the concept of XML, makes XML documents easier to localize than those created with traditional proprietary text processing or composition systems. Nevertheless, decisions made during the creation of the XML structure and during the authoring of documents can have a significant effect on the ease with which the source language text can be localized into other languages. For example, the difficulties introduced into XML documents through inappropriate use of syntactical tools can have a profound effect on translatability and cost. It may even require complete re-authoring of documents in order to make them translatable. This is worth noting, as a very high proportion of XML documents are candidates for translation into other languages.

# Designing XML Documents for Translation

It is very important to consider the implications for localization when designing an XML document. Wrong decisions can cause considerable problems for the translation process and thus increase costs. All of the following examples assume that the text to be translated is to be extracted into an intermediate form such as [XLIFF](#) (XML Localization Interchange File Format). Anyone planning to deliver an XML document directly to translators will soon be disabused of this idea after the first attempt. The intermediate format protects the original file format and guarantees that you get back a target language document equivalent to that of the original source.

An additional concept that is important regarding the localization of XML documents is the "inline" element. Inline elements are those that can exist within normal text (PCDATA - Parsable Character DATA). They do not cause a linguistic or structural break in the text being extracted, but are part of the PCDATA content.

The following is a list of guidelines based on (often bitter) experience. Most of the problems are caused by not following the fundamental principles of XML and XML best practice. It is, nevertheless, surprising how often you will come across instances of the following types of problems. Please note that this is not a proscriptive list, i.e., there may be special circumstances where the proposed rules may have to be broken:

## Avoid the Use of Specially Defined Entity References

Although entity references can look like a 'slick' technique for substituting variable text such as a model name or feature in a publication, they can cause more problems than they resolve. The following example shows how XML designers might be tempted to use entities:

```
<para>Use a &tool; to release the catch.</para>
```

### Example 1: Incorrect Use of Entity References

Entities can cause the following problems:

- *Grammatical difficulties.* If the entity represents a noun or noun phrase, this can potentially cause serious problems for languages in which nouns are strongly inflected, such as in many Slavic and Germanic languages. What appears to be correct as an entity substitution in English can cause insurmountable problems in inflected languages. The solution is to resolve all entities in the serialized version of the XML document prior to translation.
- *Parsing difficulties.* During the translation process, the text will typically be transformed into different XML-based translation formats, such as XLIFF, where the entity will cause a parsing error.
- *Problems with leveraged translation memories.* The use of specially defined entity references can also cause problems with leveraged memories. The leveraged memory may contain entities not declared in the current document.

It is generally better to use alternative techniques rather than entity references, e.g.,

```
<para>
  Use a <tool id="a1098">claw hammer</tool>
  to release the CPU retention catch.
</para>
```

### Example 2: Proposed Solution

One area where entities CAN be used to great effect is that of boilerplate text. The technique here is to use parameter entities to store the text. The text must always be linguistically complete in that it cannot rely on positional dependencies with regard to other entities, etc. Boilerplate text is used solely within a DTD (Document Type Definition). There need to be parallel target language versions of the DTD for this technique to be used. This can add to the maintenance cost, although judicious use of INCLUDE directives and DTD design can mitigate this.

### Avoid Translatable Attributes

Translatable attributes can also look like a smart way of embedding variable information in an element.

```
<para>
  Use a <tool id="a1098" name="claw hammer">
  to release the CPU retention catch.
</para>
```

### Example 3: Incorrect Use of Translatable Attributes

Unfortunately, they present the translation process with the following difficulties:

- *Grammatical difficulties.* The same problems can arise as with entity references. If you want to use the text for indexing, etc., then you cannot rely on the contents of translatable attributes to be consistent for inflected languages.
- *Flow of text difficulties.* With translatable attributes, there are two possibilities regarding the flow of text:
  - The text is part of the logical text flow.
  - The text should be treated outside of the text flow.

If the text is to be part of the text flow, then the translatable attribute causes the insertion of extra inline elements in translatable format (typically XLIFF format) of the file. If it is to be translated separately, then the translatable attribute forms a new text unit. The translator then needs to know if it is to be translated within the context of the original text unit or in isolation.

With extra inline elements, the burden is on the translator to preserve the encapsulating encoding, bearing in mind that there may be significant changes in the sequence of such

attribute text in the target language. Translation may often require that the position of the various components of a text unit be significantly rearranged.

```
<para>
  Use a <tool id="a1098">claw hammer
  to release the CPU retention catch.
</para>
```

#### Example 4: Proposed Solution

The following guideline usually applies in this case: if text has more than one word, then it should not be used in attributes. As a syntactical instrument, attributes are much more limited than elements, e.g., you can only have one attribute of a given name. The use of attributes should be reserved for single "word" values that qualify, in a meaningful way, an aspect of their element.

### Avoid the Use of CDATA Sections That May Contain Translatable Text

CDATA sections are typically used as a means of escaping multiple '<' and '&' characters. Unfortunately, they pose particular problems for tools that are extracting such text. The problem is not one of the escaped characters, but how to treat the CDATA text.

```
<TEMPLATE><<![CDATA[<p>Please refer to the
  <em>index page
  </em> page for further information</p>]]>
</TEMPLATE>
```

#### Example 5: CDATA Section Problems

The problem is similar to that posed by translatable attributes. Is the text to be treated as 'inline' to the surrounding text? Should escape sequence characters be replaced during translation with the appropriate characters that were originally escaped, or are they to be left in their escaped form? How is the software to know?

I have come across entire XML documents being embedded as CDATA within an encompassing XML document. This poses significant problems regarding the treatment of the CDATA text. It must first be extracted and then re-parsed before it can be extracted for translation.

Unless the text within CDATA sections is never to be translated, use the standard built-in character references to escape the text. Avoid using CDATA sections.

```
<TEMPLATE>
  <p>Please refer to the <em>index page
  </em> page for further information</p>
</TEMPLATE>
```

## Example 6: Proposed Solution

As an alternative, link to an external resource rather than embedding XML as CDATA:

```
<TEMPLATE xlink="ftp://ftp.xml-intl.com/res/ex1.xml"/>
```

## Example 7: Link to an External Resource

### Avoid the Use of Infinite Naming Schemes

Do not use the following type of element elm001, elm002, elm003 in well-formed documents.

```
<?xml version="1.0" ?>
<resources xml:lang="en">
  <err001>Cannot open file $1.</err001>
  <hint001>Hint: does file $1 exist.</hint001>

  <err002>Incorrect value.</err002>
  <hint002>Hint: value must be between $1 and $2.</hint002>
  <err003>Connection timeout.</err999>
  .
  .
</resources>
```

</code></pre>

<p class="figure\_cap">Example 8: Example of Infinite Naming Scheme Usage<br /> </p>

<p class="body\_1">This presents problems for extraction programs and is not regarded as

good XML practice. A much better way of doing this is to use the ID and IDREF attribute

mechanisms to link elements together.</p>

<pre>

```
<code> <?xml version="1.0" ?>

<resources xml:lang="en">
  <error id="001">
    <caption>Cannot open file $1.</caption>
    <hint>Does file $1 exist.</hint>
  </error>

  <error id="002">
    <caption>Incorrect value.</caption>
    <hint>Value must be between $1 and $2.</hint>
  </error>
  .
  .
</resources>
```

## **Example 8: Proposed Solution**

In part 2 of this series, Zydron will describe nine more points for properly localizing XML, including ways to handle graphics, dealing with text expansion, and properly marking up text to facilitate localization in languages with widely varying typographical conventions.