**STEPBible: Musings on markup**

Musings on markup



**Contents**

[1 Introduction 3](#_Toc152068380)

[2 Purposes of markup 3](#_Toc152068381)

[3 Mechanics of markup 4](#_Toc152068382)

[4 The aims of our processing 4](#_Toc152068383)

[5 Reading mode 5](#_Toc152068384)

[6 Study mode 6](#_Toc152068385)

[7 Some problems 6](#_Toc152068386)

# Introduction

Latterly we’ve been struggling rather with formatting and markup. In fact the issues we’ve been facing aren’t new: we’ve been aware of them for a long time, but they’ve rather dropped off our radar.

In this document I take a look at the issues and try to suggest an alternative data representation which might help avoid these problems. If I manage to do this, though, I have to admit that the result will almost certainly be entirely impractical: we’re too tied into Sword modules to be able to contemplate a radical alternative. But at least perhaps the discussion will stimulate debate and perhaps help us to develop some more practical approaches.

Before we start, I should probably say that what follows will be conditioned to some extent by the process with which I am most familiar – that of taking a USX representation of a text[[1]](#footnote-1), converting it to OSIS and then using osis2mod to create modules. Where the starting point is OSIS (or even where it is an existing module), we may have rather less flexibility.

# Purposes of markup

Both USX and OSIS are XML-based markup languages. There is, unfortunately, no straightforward 1:1 mapping between the two of them, but they do at least have similar philosophies. For the following discussion it will be useful to divide the tags they contain into a number of major categories:

* **Aesthetic**: Tags which are intended purely to prettify the output.
* **Readabilty**: Tags which are used mainly to make it easier to read the output (for example by increasing the amount of whitespace on a page in order to make it appear less daunting).
* **Formatting**: Tags which are there to alter the formatting in a way deemed useful to the understanding of the reader. For example tags which directly italicise certain portions of the text.
* **Semantic**: Tags which reflect the underlying meaning of the text. For example a tag which marks a number of words as having been added to the canonical text to make the translation flow more smoothly.
* **Functional**: Tags which, in an electronic text, might well be expected to invoke some kind of function – eg a footnote tag which may cause the footnote to be displayed in a pop-up.
* **Structural**: Tags which reflect the structure of the text – book, chapter and verse divisions etc.

We can simplify this somewhat. (Make the most of this – there’s precious little opportunity to simplify things …)

I believe we can ignore tags aimed purely at aesthetics and readability because we simply shouldn’t be getting any. Translators have no control over this aspect of things, because they cannot know how the text will actually be rendered; and we don’t tend to worry about such things in STEPBible. (That’s not to say that we’re happy to produce stuff which looks awful: just that we don’t rely upon the text itself to tell us how to make things look pretty.)

And I think that by the time we start to get close to the rendering, the distinction between semantic and formatting has largely disappeared. We reveal the semantics to the user through the formatting we apply; and the formatting we apply usually has semantic implications. Even something as simple as a paragraph boundary is there to suggest that there is a change of topic in the underlying text.

# Mechanics of markup

In terms of how the markup actually works, it’s useful to split it up into the following groups:

* Enclosing tags: To state the blindingly obvious, these are tags which enclose other tags and text; and for our purposes, we can further limit this in that we’re really interested only in canonical text. There will be enclosing tags which enclose non-canonical text such as footnotes, but those aren’t really enough of a problem to require discussion here.
* Self-closing tags: A tag with no contained text, such as a line-break marker.
* Milestone tags: A tag used to mark a particular position in the text. For our purposes these usually come in pairs – for example one to mark the start of a verse and one to mark the end (often referred to – by me at least – as *sid* and *eid* markers, because the start marker usually has a **s**tart **id**, and the end marker usually has an **e**nd **id**). Milestone tags are an alternative to what you might think would be enclosing tags (ie you’d normally expect a verse to lie entirely within an enclosing tag). They have been introduced specifically so that formatting / semantic tags can run across verse boundaries. And the fact that they exist is a major complication as far as processing goes.

# The aims of our processing

Things are complicated by the fact that we use a given text for two significantly different purposes. We use it essentially in ‘reading’ mode, where we are basically presenting an electronic version of a printed text, albeit one with useful added features like clickable cross-references, Strong’s etc. And we also use it in ‘study’ mode, where we are presenting specific verses or collections of verses on their own in response to requests for interlinear display[[2]](#footnote-2), searches, etc.

These two modes differ very significantly: in reading mode we are seeing verses within their context, whereas in study mode we are seeing them essentially out of context. I’ll return in a moment to the implications of this difference.

Support for both reading mode and study mode constitute two of our aims. The third aim is that where licensing terms permit, we also want to produce modules which would be of use to third parties. This gives a constraint upon what we can do. Recently, for instance, we have been looking at things we might do in OSIS which would enable us to produce a better rendered text in STEPBible. But these things would have created modules which would, to all intents and purposes, been useless to third parties.

Returning to our own issue of reading vs study mode, I observed earlier that milestone markup is very often used with verses, and that it is used specifically to permit formatting to run across verse boundaries (and therefore *encourages* it to do so – and indeed most of the texts I’ve seen use cross-boundary markup).

When looking at things in context (particularly if the context is an entire chapter) this is never an issue, because all of the verses we are looking at sit entirely within the context, so there is no clash between formatting and verses.[[3]](#footnote-3) When looking at things out of context, life is much more complicated. We frequently need to extract the content of an individual verse, and that can be very difficult if there is markup which runs across the verse boundaries – how do you extract the relevant text, and what formatting do you apply to it?

In fact, reading mode and study mode differ so significantly that it would really warrant having two different representations of each text, one for each mode. The problem with that, though, is that it would significantly increase bandwidth and storage requirements. The trick is to come up with a single representation which can meet both needs.

# Reading mode

The essential requirement in reading mode is that the text should be presented in ‘exactly’ the way specified by the translators – both because this is what users will be familiar with and also because licensing conditions often demand it.

I put the word ‘exactly’ in quotes there because in fact we often don’t know precisely what the translators require: they may, for instance, give us semantic markup indicating that certain words have been added to the canonical text to make it read more smoothly, but they don’t tell us how they expect that text to be rendered (but see next paragraph). And in some cases, they can’t use the ‘correct’ markup anyway. One particular example of this – particularly common and particularly problematical – is indented text. Neither USX nor OSIS actually have a markup which corresponds to this, so translators often have recourse to bullet-point list markup in the hope that this might actually end up being rendered as indented text without the bullet points.

I said above that with some forms of markup we may well not know what the translators had in mind by way of rendering anyway. In fact with texts from DBL, we normally *could* know, because texts are supplied with a styles.xml file which tells you. However, presently we have no way of taking this into account – we apply exactly the same styles to all texts.

I see this as a significant drop-off, since, for example, it potentially leaves us applying italics to sections of text which are written using a script which does not support italics, or where particular types of formatting carry with them a semantic load at odds with the way we see that formatting in the West. Or take NASB, which goes to appreciable lengths in the NT to identify OT quotes, and I think states in the accompanying explanatory material that these quotes are italicised. Unless we have the ability to apply styles appropriate to this particular text, there is, in fact, no guarantee that these passages *will* be italicised.

# Study mode

As I suggested above, the essential issue with study mode is to be able to extract verses for individual display, perhaps for use in interlinear mode, for use as the targets of cross-references, or as search results.

Even to identify the relevant text can be complicated when parts of it may fall under one tag, and other parts under another, and when bits of it may be nested withing subtags and bits not. But there then remains also the issue about what you do with the ‘broken’ formatting – if, for example, you have a ‘translator’s addition’ tag which starts outside the verse and ends inside it.

# Some problems

A problem we have encountered recently probably serves to illustrate the issues as well as any. The translators had used para:q for poetry, and were, I think, reliant upon this generating fully indented paras (slightly indented for para:q, more indented for para:q2, etc).

What was actually happening was that para:q gave you a small-ish first-line indent and then reverted to the left margin. para:q2 gave you a larger first-line indent, before it, too, reverted to the left margin … and so on.

So why? After all, para:q is an enclosing tag in USX – it contains a line of poetry – and that would seem to give you ample opportunity to make it work properly.

A glance at the generated HTML gives you a clue. para:q is a para tag in USX, and you might reasonably expect that para tags end up as div’s in HTML – and, moreover, div’s which contain the text which the para:q contained. But what is actually generated is a <span>, and, moreover, an *empty* span, positioned immediately before the text you might reasonably have thought it would contain. This, of course, explains the visual behaviour. The empty span can generate a first line indent, but it cannot apply formatting to its supposed content, because its supposed content is, in fact, outside of it.

Which in turn raises the question of why it should have been treated in this way. And I am 99% certain that the answer is that turning it into a self-closing tag avoids any possible cross-boundary issues, and therefore makes it easy to support study mode. Were it not self-closing, there is a high chance that you would, somewhere or other, have cross-boundary markup, and study mode would then break.

The change, therefore, supports study mode. But it breaks reading mode, because it is simply not possible to produce the indented paragraphs which the translators would have anticipated, and therefore not possible to make the text look similar to a printed version of the same Bible.[[4]](#footnote-4)

\* End of document \*

1. In this I also include VL (VerseLine) input, since that can be seen as proceeding via USX. [↑](#footnote-ref-1)
2. And yes, they are on their own even in interlinear display because although we may be displaying a lot of related verses on the same screen, two or more different sources are interleaved, so each individual verse is isolated from the other verses in the same text by the interposition of verses from the other text. [↑](#footnote-ref-2)
3. Tags never run across chapter boundaries. More strictly, formatting and semantic tags never run across chapter boundaries, and the only higher level structural tag – the book tag – contains the chapter in its entirety. As a consequence, there is no tag which starts outside of a chapter and ends inside it, or starts inside it and ends outside it. [↑](#footnote-ref-3)
4. There is a similar issue with para:p. In USX this is often presented in enclosing form, but osis2mod converts it to a self-closing tag which, in the module or in the rendered HTML, is placed before the text which ‘ought’ to form its body. Again the result is that if you wanted to apply some kind of style to the content of the paragraph as a whole, you are thwarted. [↑](#footnote-ref-4)