**Formatting in STEPBible  
or  
The impossibility of indentation**

Formatting in STEPBible



**Contents**

[1 Introduction 3](#_Toc152592544)

[2 The mechanics 3](#_Toc152592545)

[3 The issues 4](#_Toc152592546)

[3.1 Sometimes we don’t even do what *we* want 4](#_Toc152592547)

[3.2 Sometimes we DON’T do what the translators want 4](#_Toc152592548)

[3.3 Sometimes we CAN’T do what the translators want 5](#_Toc152592549)

# Introduction

We can’t do indented paragraphs.[[1]](#footnote-1) Every so often, that troubles us. Then other things bubble to the top of the agenda and we forget about the problem … until the next time. This is one of those times.

In fact, the problem goes slightly wider than that, and I think can best be expressed as:

1. We *don’t* do what the translators want.
2. We *can’t* do what the translators want.
3. Sometimes we don’t even do what *we* want.

All of which I look at in more detail in section 3 (albeit not in this order).

A word of warning, though. Regrettably, I don’t think there’s an easy solution – or not to all of the issues. I don’t even think there’s a *difficult* solution. There may perhaps be a *very* difficult one, but even then I’m not convinced. But perhaps by discussing the issue it may help provoke some alternative ideas.

# The mechanics

As a reminder, the bit of the process we are concerned with here normally runs something like this:

|  |
| --- |
| ***Rendered output***  **Sword module**  **Style**  **HTML**  **JSword /  STEP**  **OSIS**  **osis2mod** |

where the shaded boxes represent data formats, and the non-shaded ones represent transformation processes.

We have no control over osis2mod[[2]](#footnote-2), nor over JSword2, and I am not sure of the extent of any processing of our own in the JSword / STEP step. The one thing we *do* have control over (although at present no one seems to recall how) is the CSS stylesheet at the end of the process.

So we are reliant upon a process over which we have at best limited control – reliant upon it to produce sensibly structured HTML which aligns with our stylesheet.

# The issues

## Sometimes we don’t even do what *we* want

I’ll deal with this first because it is (probably) the most tractable issue.

There are certain things (at present only acrostic headings, but there are probably other things we have yet to identify) which are not being formatted as we wish, and which could almost certainly be fixed if we were to make simple changes to the CSS stylesheet.

Currently in order to get round this, I am generating the ‘wrong’ OSIS – OSIS which produces the effect we want, but which loses the semantic information which would be present if I had not intervened. (More specifically, as I recall, we want the acrostic headings to be italicised and in bold face. There is a proper way in OSIS of marking an acrostic header, but our stylesheet does not respond to the HTML generated for that tag. I am therefore not generating the acrostic heading tag; rather, I am generating italic and boldface tags.)

The problem with this workaround is precisely that it does lose semantic information. If we are using modules only within STEP, then this may not matter hugely. But if we hope to make them available to third parties, they will be less useful without the semantics.

This should be an easy thing to address – and indeed I believe we have tweaked the stylesheet in the past to handle precisely this kind of issue. The main problem at the moment is that no one seems to know how to track down the master stylesheet and how to ensure that changes to it actually feed through into the rendering.

## Sometimes we DON’T do what the translators want

At least with texts from DBL (and most of the texts I am working with at present do come from that source), there is a mechanism (a styles.xml file) whereby the translators can indicate to us how the various USX tags should be rendered.[[3]](#footnote-3) We make no use of this: all texts are rendered using a single built-in stylesheet of our own devising.

*Can* we do better? I’m not sure. Certainly in theory we have the necessary information available, at least on DBL texts. But I have no idea whether we have a mechanism for passing this information through to STEPBible.

*Should* we do better? Maybe, maybe not. Clearly to handle per-text stylesheets like this represents an additional complexity which we might prefer to avoid. And at least as things stand, the single fixed stylesheet does mean that we have a house style. If text marked as having been added by the translators is italicised in one Bible, it will be italicised in all.

On the other hand, it is perhaps not entirely clear that we have the remit to do our own thing – licensing conditions may preclude it. And whether we do or not, changing the formatting is something which perhaps needs to be undertaken carefully. As I recall there is one Bible (NASB, is it?), which goes to great lengths in the NT to format OT quotations distinctively – and possibly states as much in the information we transfer to the copyright page. If they say that OT quotes are italicised throughout, it rather behoves us to ensure they *are* italicised.

There is another reason, too, why it may be desirable to work with formatting requirements as expressed by the translators: I am not at all sure that all scripts support the text decoration techniques we commonly use on Roman scripts (nor actually that even if supported, they necessarily carry the same semantics as in western languages). Consistently italicising a particular element of a text is all well and good, but only so long as the target script actually *supports* italics. The translators, much more than we, are likely to know what is appropriate and will work.

## Sometimes we CAN’T do what the translators want[[4]](#footnote-4)

Here we come back, for instance, to indented paras. Very often the translators want them. Very often *we* want them, if for no other reason than because the translators want them. And we can’t do it.

This, I suspect, is the most intractable of the three problems; and it comes down, I believe, to the fact that we are trying to support two entirely disparate things using a single data representation.

On the one hand, we want to be able to work in ‘reading book mode’, where we are dealing with large chunks of text in context (entire chapters, say), and we want the display to be essentially just an electronic version of a printed book[[5]](#footnote-5).

And on the other hand, we need to support ‘study mode’, in which we need to be able to identify and display very small pieces of text out of context, perhaps as small as an individual verse (eg for use as a search result, or in an interlinear display). And while on the face of it these two requirements are very similar, they are in fact so different as to be seriously at odds with one another.

Considered stand-alone, book mode is easy. Something gives us HTML, we style it a little bit, and then we display it. In book mode, all that really matters is the formatting. Verses are somewhat of an irrelevance: all they do is turn up in the output as little superscripted numbers effectively as part of the canonical text.

Considered stand-alone, study mode is relatively easy too (depending upon how sophisticated we need to be). We just have some kind of database which maps scripture references to canonical text, and then we display that text. Here, all that really matters is the verse structure. Formatting is simply a complication (albeit a distinctly complicated complication).

But the problem is that we’re *not* handling either of these things stand-alone. To reduce network bandwidth and storage requirements, we have a representation which attempts to support both; and in fact, in some measure, it supports neither.

The huge difficulty is the fact that both USX and OSIS support markup which runs across verse-boundaries (and in supporting it, encourage it: I don’t think I have yet seen a text in either of these two representations which does not use it). But cross-boundary markup is a reading-book-mode thing; it’s not a study-mode thing. In study mode, where you need to be able to isolate the content of individual verses, it’s a complete pain, because you now have to confront the fact that the content of a single verse is no longer going to be contiguous: quite possibly different bits of it are going to fall under a number of different parent tags.

You are then faced with two issues. First, how can you manage to gather together all of the parts of the verse; and second, how are you going to format them, given that different parts may fall under different parents, and each parent may impose its own formatting? In fact, I’ll ignore the formatting issue here. To do so skates over some very complicated issues, but I think we have bigger fish to fry – or certainly fish which are easier to describe.

How to gather together all the parts of the verse, then? So far as I can see, osis2mod addresses this simply by avoiding the issue altogether. And it does *that* by ensuring that there *is* no cross-boundary markup.

In support of this thesis, I can point to the fact that USX’s para:p (which normally *encloses* text) ends up, by the time we get as far as HTML, as an *empty* tag positioned *before* the text which it previously enclosed. Similarly para:q (used for poetry) ends up as an empty span tag before its text. And with complicated structures, occasionally osis2mod simply gives up and announces that the OSIS (perfectly *valid* OSIS) is in fact badly structured and beyond its capabilities.

The basic approach encapsulated in the Crosswire software, therefore, appears to be that any *enclosing* div-flavour tag is replaced by an empty tag, positioned at the start of where the div is supposed to appear. (Span-flavour tags are probably less of an issue – for reasons I won’t go into here, they can normally be manipulated fairly readily to avoid them running across verse boundaries.)

Doing this, of course, guarantees that all cross-boundary tag issues go away (at least in respect of the tags where osis2mod feels able to make this kind of change). The trouble is that collapsing div-flavour tags like this is an entirely illegitimate thing to do. An enclosing div can apply formatting to its content (like indenting it, for instance). A single-point div cannot because it *has* no content.

And this is the situation we are facing. By the time the text gets as far as being rendered, text blocks which we would like to indent (or whatever) are no longer contained within any tag to which we can apply indentation: we simply have no relevant information. (This is why the suggestion that we need only apply the appropriate style settings does not work – there is nothing to which we *can* apply it.)

Assuming my analysis of the operation of osis2mod and JSword is correct here – and I believe it is – then the only way we can address this (ie the only way we’re ever going to get things like indented paras etc) is by completely changing the way osis2mod and JSword work. But that, of course, is a huge change – and one made hugely more complicated by the need to carry on supporting modules produced under the existing regime. Other ideas would be gratefully received!

\* End of document \*

1. It’s not often I get to do a footnote this early. Indented paras are a significant motivation for this document, so it might be as well to know what we’re talking about here. By an indented para I mean not one with an indented first line: I mean one with an increased left and / or right margin throughout. [↑](#footnote-ref-1)
2. Except insofar as we do actually have our own version of the software now. [↑](#footnote-ref-2)
3. Admittedly I don’t know how much effective use is made of this. It’s perfectly possible that DBL have some standard default file for this, and people may simply use a copy of this file, untailored. [↑](#footnote-ref-3)
4. This section is slightly speculative, because I am not party to the internals of osis2mod and JSword, and therefore am reduced to guessing how they work based upon the outputs they produce. [↑](#footnote-ref-4)
5. And indeed, with the Biblica licensing conditions, are *required* to make it look like that. [↑](#footnote-ref-5)