Jons converter

Jon’s Word-to-SC converter

Jon Rosewell, Feb 2025

# Introduction{id="first"}

## Purpose

NB Work in progress...

My converter is intended to enable structured authoring in Word of material destined for the VLE (and possibly for print). It provides a (relatively) quick preview via oXygen. To preview:

1. copy some or all of the Word document
2. paste into a skeleton document in oXygen
3. if necessary, attend to any errors shown
4. click Render in the StructuredContent menu or toolbar
5. if appropriate, click the Save link to publish on the VLE
6. go home early ;-)

The Word document must be carefully styled using style names that correspond to OU structured content (SC) tags (element names); see the [Structured Content Tag Guide](https://learn3.open.ac.uk/mod/oucontent/view.php?id=185747) for the complete list. To author from scratch, I can supply a Word document empty-sc.docx containing the required styles. To author from existing OU web material, I can convert the master XML file back to appropriately styled Word.

The visual appearance of these documents once in Word should be reasonably comfortable to use; if not, the font face, size, colours etc of styles can be freely tweaked without detriment since the conversion relies only on the style name. Note that the list of SC elements and therefore styles is long: use the Word Styles pane to see all styles; right-click and Add to Style Gallery those you use frequently.

This document serves both as documentation and as an example / test document – try it out!

Software can be found on GitHub <https://github.com/JonRosewell/struct-auth>.

# Basic text

This is a paragraph with some direct character formatting for: **bold**, *italic*, underline, sub/sup, small caps, **strong**, *emphasis*, created with standard Word buttons.

And some standard OU character styles: ComputerCode, ComputerUI, GlossaryTerm.

Paragraphs can contain author comments and editor comments.

Paragraphs can contain several sorts of link. For example, an external hyperlink to [Google](https://www.google.com/) can be created in Word in the normal way. For references to other documents or parts of the website, it is better to use an <olink> which refers to a document and section names, for example to the [Course guide](Course Guide#11). Use <CrossRef> to refer to the [next session](lists) or other parts of the *same* document; this requires placing ids on destinations. See the SC reference about [links](https://learn3.open.ac.uk/mod/oucontent/view.php?id=185750&section=6).

A paragraph can contain an equation such as e = m c^2 inline. Equations can also be shown as display:

\left(x+a\right)^n=\sum\_{k=0}^{n}{\binom{n}{k}x^ka^{n-k}}

For more detail, see [Equations](equations).

Program listings can be created:

pi@raspberry:~ $ mkdir test  
pi@raspberry:~ $ cd test  
pi@raspberry:~/test $ pwd  
/home/pi/test  
pi@raspberry:~/test $

In a program listing, whitespace is (or should be!) preserved so indentation can be achieved, but it is safest to use non-breaking space! There is no syntax highlighting but a bit of bold and italic can help.

XSLT is powerful: this is what is required to do automatic tagging:

<!-- default: para with class (= Word para style) or span with class (= Word char style) become an element of same name -->  
<**xsl:template***mode*="styling" *match*="h:p[@class] | h:span[@class]" *priority*="0.4">  
    <**xsl:element** *name*="{@class}">   
        <**xsl:apply-templates** *mode*="styling"/>  
    </xsl:element>  
</xsl:template>

## Equations{id="equations"}

Basic approach to equations is to paste TeX code and style, either as *Equation* or *InlineEquation*, which will translate to <Equation> and <InlineEquation> respectively.

This is a sample TeX equation styled as *Equation*:

z^2=\frac{\text{GDP}}{(r+1)^2}

And the same as an inline equation z^2=\frac{\text{GDP}}{(r+1)^2} in the middle of a paragraph.

TeX code can be copied from existing VLE pages. The VLE will display equations either as generated image or better-quality client-side MathJax (choose in your VLE profile settings). Both have accessibility features (click or right-click) that include showing TeX equivalent, so use that to grab TeX for existing equations. Paste TeX code as styled line to transfer to SC XML.

To also see a rendered version in Word, you can insert a new Word equation, paste TeX code into it *as text* and press Return to render. Switch between Professional and Linear to show fully rendered or TeX source.

If you create or edit an equation using Word Equation Editor, copy/paste it *as text* to a styled line. If a pasted equation doesn’t produce TeX, change Word Equation Options, choosing Copy Linear Format to clipboard as plain text.

Examples: an equation inserted using Word equation editor inline \frac{a}{b} with *InlineEquation* version following. Then an equation inserted as display equation in Word and then copied as *Equation*:

\left(x+a\right)^n=\sum\_{k=0}^{n}{\binom{n}{k}x^ka^{n-k}}

*Limitation*: The Word equations are lost in conversion but the TeX versions are passed to structured content. Unfortunately it is a manual job to keep them in sync.

*Gotcha*: If you encounter an error 'Warning: The image source (src attribute) has not been specified' when rendering, this may relate to an empty *InlineEquation*. The culprit may be an end-of-line character in Word that is styled as TeX, most likely within a list. Highlight the end-of-line and press Ctrl+Space.

Note 1: I don’t know what current practice is for editors handling maths, but I believe it used to involve a MathFlow plugin, both in Word and oXygen. This produces MathML, as can Word Equation Editor. This can also be passed through to SC VLE instead of TeX but it is verbose and unreadable so I find TeX a better alternative.

Note 2: If text in equations differs in size compared to body text, the underlying reason is the VLE handling of high resolution displays. For possibly good reasons, this fudges between the actual hardware pixel size and a nominal web CSS ‘pixel’ and this fudge may not be accurate.

Note 3: The same issue with high resolution displays can make equations appear fuzzy compared to body text of a similar size: equations have been rendered as low resolution JPEG files. You can switch to better client-side MathJax rendering using your VLE profile; unfortunately the default remains JPEG for all users.

## Numbering{id="numbering"}

Numbering the really easy way

The simplest approach is to do no numbering whatsoever in your Word document, convert your document to XML, and then run my numbering script (number-sc.xsl). This will number your document – Sessions / Sections / SubSections, Figures, Tables – in OU house style. You may then go home early.

I recommend that in text you only refer to ‘the figure below’ or ‘the figure above’ rather than ‘Figure 9’. If absolutely essential, refer to ‘Figure X’ and then search and manually fix references after automatic numbering. For distant references you should in any case use a <CrossRef> link with the destination title (not numbered).

A downside is that your Word document will remain unnumbered but you will soon get used to that ;-)

In detail, the numbering transform does the following:

* Strips any existing <Number> tags from previous numbering
* Places <Number> at the start of <Title> for Session/Section/SubSection, <Figure><Caption>, and <Table><TableHead>
* Inserts a number in outline style for sections, and in simple sequence for figures and tables.

*Limitation*: The numbering script doesn’t currently number activities since I don’t know how best to deal with the many variants. But it can be easily added once decided whether Activities, Exercises, SAQs, ITQs... should be numbered in a single or several sequences.

*Limitation*: Equations probably have to be manually numbered, at least to use explicit tag/style (actually *Label* rather than *Number*) because typically only some steps are numbered, but could perhaps be renumbered automatically to cope with changes. References would still break, though, so perhaps this is a case for using Word SEQ numbering described next?

See later to [set up script](setupscripts) for numbering.

Numbering the less easy way

If you find it essential to number your Word document as you write, then you can insert numbering fields which Word will update automatically. On conversion to XML, these will be converted to literal text. With the appropriate use of fields, you can number figures, tables, and headings in (more or less) OU house style.

* To insert a Word field, use toolbar Insert > Quick parts > Field and choose SEQ. Give an identifier to keep sequences separate, for example use ‘Figures’ to get {SEQ Figures} and similarly for ‘Tables’ and ‘Activities’.
* To achieve OU style outline numbering for headings, insert Insert > Quick parts > Field and choose LISTNUM with option LegalDefault, ie {LISTNUM LegalDefault}.
* To show existing Word fields, select all Ctrl+A and Shift+F9. F9 updates a field; select all Ctrl+A to update the whole document.

Copy an existing figure caption or heading to make life somewhat easier!

*Limitation*: Word fields are lost when a document is converted to XML, so round-tripping converts auto to manual numbering.

*Limitation*: Strictly, figure and table numbers should be surrounded by a <Number> tag, and appearance improves if this is done. Heading numbers should also be wrapped in <Number>; I believe this has no visual effect on the VLE, although it does in print. Practice seems to vary. The script will do it for you ;-)

Numbering another easy way

The automatic numbering features of Word *styles* are easy to set up and will give you numbering when authoring. However, the numbers do not transfer as text and will be lost on conversion. Subsequent use of my numbering script will provide numbers; this numbering should be identical but I can’t guarantee that will always be the case. Should be worth the gamble...

## Tables

Simples: just use normal Word table features, and things will work...

Simple table{style="topbottomrules"}

|  |  |  |  |
| --- | --- | --- | --- |
| **Header** | **Col one** | **Col two** | **Col three** |
| Row 1 | row 1, col one | row 1, col two | row 1, col three |
| Row 2 | row 2, col one | row 2, col two | row 2, col three |

Plain vanilla tables will work, including left/centre/right alignment (use Paragraph > Align toolbar buttons) and decimal alignment (use Paragraph > Tabs > Decimal).

*Limitation*: alignment requires heuristics; let me know if unreliable. Decimal alignment likely to be lost on round-trip conversions; recreate decimal tab stop if necessary.

Cell alignment

|  |  |  |
| --- | --- | --- |
| **Header** | **Text in a wide column** | **Decimal in a wide column** |
| Row 1 | Left | 1.23 |
| Row 2 | Centre | 123.4567 |
| Row 3 | Right | 123.0 |

Merging cells works, both horizontally and vertically.

Cells merged horizontally

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Header** | **One** | **Two** | **Three** | **Four** |
| Row 1 | Merging cells horizontally means text spills across | | Row 1 | Row 1 |
| Row 2 | Row 2 | Row 2 | Row 2 | Row 2 |

Cells merged vertically

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Header** | **One** | **Two** | **Three** | **Four** |
| Row 1 | Merging cells vertically means text spills down if it is long enough to fill the cell. Merging cells vertically means text spills down if it is long enough to fill the cell. Merging cells vertically means text spills down if it is long enough to fill the cell. Merging cells vertically means text spills down if it is long enough to fill the cell. | Row 1 | Row 1 | Row 1 |
| Row 2 | Row 2 | Row 2 | Row 2 |
| Row 3 | Row 3 | Row 3 | Row 3 | Row 3 |
| Row 4 | Row 4 | Row 4 | Row 4 | Row 4 |

*Limitation*: SC tables (like HTML) can have header rows/columns which are tagged and rendered differently to other cells. I have no equivalent in Word; using bold text is probably the best that can be easily achieved. Can be fixed manually in oXygen at handover.

*Limitation*: Complex styling is not going to work: SC offers limited styling and most of this has no effect on the VLE anyway, and I have made no attempt to implement mapping from Word to SC except as above. For a complex layout, I suggest you insert a placeholder image of your desired layout and hope someone can replicate something close in oXygen at final handover.

# Lists{id="lists"}

Structured content has three types of list.

Bulleted list:

* Item one
* Item two

Numbered list:

1. Item one
2. Item two

Unnumbered list:

Item one

Item two

To create lists in Word, use the toolbar buttons for bulleted, numbered and outline lists. Use the *List Paragraph* style to create an SC <UnNumberedList>.

*Care needed*: In general, Word allows you to mess with indentation to create things that look like lists, but the result may simply be indented paragraphs and the indentation will be lost on conversion. To remedy problems, try:

1. Remove list styling using toolbar buttons, or
2. Remove list styling by applying *Normal* style, or
3. Remove all styling with Styles > Clear all.

Then reapply list structure cleanly.

Nested lists are possible, although structured content only allows two levels.

Bulleted list:

* Fruit
  + Apple
  + Banana
* Vegetables
  + Aubergine
  + Broccoli

Numbered list:

1. Fruit
   1. Apple
   2. Banana
2. Vegetables
   1. Aubergine
   2. Broccoli

An outline / nested list where numbering style differs at each level is possible in both Word and SC, but conversion is not guaranteed. Use the Word outline list toolbar button.

1. Fruit
   1. Apple
   2. Banana
2. Vegetables
   1. Aubergine
   2. Broccoli

Numbered lists don’t have to start at ‘1’ (or ‘a’, ‘i’); you may wish to continue a list after intervening content. First an alphabetical list starting at (d):

1. the
2. cat
3. sat

A lower-roman list starting at (v):

1. on
2. the
3. toadstool

*Limitation*: for lists created in Word, the first digit is used to infer the list type and start number. So some start values are ambiguous (i, v, o, x) or not possible (numbers beyond 9, letters beyond z). You can fix once in XML.

The examples above are simple: each list item is only a single paragraph. These are easy to create in Word and will translate straightforwardly to structured content. See the next section for dealing with more complex list structures.

## Extended lists

Structured content allows list items to have extended content: more than one paragraph of text, figures, tables, equations etc. This is straightforward in SC since the tagging shows where each item begins and ends. However, where a Word list item has extended content, the list is broken around the additional content, and this would lead to poor structure in SC.

For simple cases, inserting Word manual line breaks (Shift+Enter) within a list item is sufficient to achieve the desired effect:

1. 1. The run is 3-1.5=1.5. The rise is 2-6=-4. So the gradient is  
      \dfrac{\text{rise}}{\text{run}}=\dfrac{-4}{1.5}=\dfrac{-40}{15}=-\tfrac{8}{3}.  
      (Check: The line slopes down, so the gradient should be negative.)
   2. The run is 1-(-1.5)=1+1.5=2.5. The rise is 2-(-2)=4.  So the gradient is  
      \dfrac{\text{rise}}{\text{run} }=\dfrac{4}{2.5}=1.6.   
      (Check: The line slopes up, so the gradient should be positive.)
2. The rise between any two points on a horizontal line is zero.   
   Because the gradient is the rise divided by the run, it follows that the gradient of a horizontal line is zero.

This hack isn’t possible if the extended content includes figures, tables, display equations and so on. In that case, surround the list with blank lines of *ListHead* and *ListEnd* style. Within this pair, any content following a list item is assumed to be part of the same item until the next item is reached. (Note that indentation in Word is ignored when converting to SC.)

 [list head]

1. 1. The run is 3-1.5=1.5. The rise is 2-6=-4. So the gradient is

\dfrac{\text{rise}}{\text{run}}=\dfrac{-4}{1.5}=\dfrac{-40}{15}=-\tfrac{8}{3}.

(Check: The line slopes down, so the gradient should be negative.)

* 1. The run is 1-(-1.5)=1+1.5=2.5. The rise is 2-(-2)=4.  So the gradient is

\dfrac{\text{rise}}{\text{run} }=\dfrac{4}{2.5}=1.6.

(Check: The line slopes up, so the gradient should be positive.)

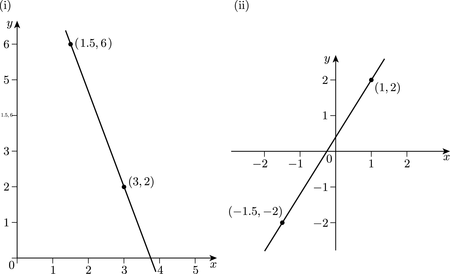
1. The rise between any two points on a horizontal line is zero. Because the gradient is the rise divided by the run, it follows that the gradient of a horizontal line is zero.

 [list end]

In a few cases, a further bracketing may be required around a sublist to clarify the structure:

 [list head]

1. Calculate the gradients of the lines shown below.



\\dog\units\_non\_course\TM110\Unit 6\mu123\_u6\_fact\_05.png

1. For each of the following pairs of points, calculate the gradient of the line that passes through them.
   1. A(2,2) and B(-2,2)
   2. A(2,2) and C(-1.5,-2.5)
   3. A(2,2) and D(3,-1)

(You might find it helpful to first sketch these points on a graph.) SubListHead/End clarifies that this line belongs to b) not iii)

1. What is the rise between any two points on a *horizontal* line? What is the gradient of a horizontal line?

 [list end]

If all else fails, replace the first level of listing by *SubHeadings* and (if necessary) the second level by *SubSubHeadings*. (Note: *SubHeading* and *SubSubHeading* can only be used within an activity or box, but you can insert an *InternalSection* in body text to avoid this restriction.) Doing things by hand is also a way to escape the SC restriction to two list levels.

Question 1

(a)

1. The run is 3-1.5=1.5. The rise is 2-6=-4. So the gradient is

\dfrac{\text{rise}}{\text{run}}=\dfrac{-4}{1.5}=\dfrac{-40}{15}=-\tfrac{8}{3}.

(Check: The line slopes down, so the gradient should be negative.)

1. The run is 1-(-1.5)=1+1.5=2.5. The rise is 2-(-2)=4.  So the gradient is

\dfrac{\text{rise}}{\text{run} }=\dfrac{4}{2.5}=1.6.

(Check: The line slopes up, so the gradient should be positive.)

(b)

The rise between any two points on a horizontal line is zero. Because the gradient is the rise divided by the run, it follows that the gradient of a horizontal line is zero.

# Boxes{id="boxes"}

There is a variety of box-like structures available in structured content. Most were intended to have semantic meaning, for example using <Reading> means you could generate a list of all the Readings in a module, but there is also a generic <Box>. All function as containers, both structurally and visually. In Word, a box is created between pairs of styled paragraphs, for example *StudyNoteHead* and *StudyNoteEnd* to enclose a <StudyNote>. Any text on the *StudyNoteHead* line will be used as a heading; text on the *StudyNoteEnd* line will be lost.

If you don’t follow these rules, the generated XML will show validation errors. Hopefully these will help you understand the problem and fix it in XML; you can then correct the styling of your Word master so that the next conversion is less painful.

Example

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Extract

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Reading

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Study note

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Case study

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Box style 1{type="style1"}

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Box style 2{type="style2"}

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Box style 3{type="style3"}

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

Box style 4{type="style4"}

A paragraph of body text.

A paragraph of body text.

A paragraph of body text.

A study note doesn’t require a heading.

A paragraph of body text.

All types of box don’t require a heading.

A paragraph of body text.

Icons{resource1="reading\_matter"}{resource2="calculator"}{resource3="forum"}

All boxy things can be flagged with up to three icons from a [long list](https://learn3.open.ac.uk/mod/oucontent/view.php?id=185755&section=1.1).

A paragraph of body text.

Quote heading

This is a quote

Another quoted paragraph.

Anon (traditional) <Reference>

A paragraph of body text.

An <InternalSection> acts like a box but is considered later in [Test of heading levels](test-headings).

A paragraph of body text.

Limitations

Although SC allows it, I do not support nested boxes in any permutation. On the VLE, nothing much is achieved by doing so anyway – only one structure is visible. Remember that all boxes allow *SubHeading* and *SubSubHeading* if you need some internal structure.

A paragraph of body text.

Key points

First important thing: remember to use a matching closing style, e.g. *BoxHead* and *BoxEnd*.

Second important thing: for a box without a heading, use an empty *BoxHead*.

Final thing: A <KeyPoint> looks just the same as a normal para to me.

KeyPoints are structured differently and have a different semantic import to other boxes but are visually pretty similar (shame).

# Activities

Structured content supports a range of structures for activities of various sorts. <Activity> is probably the generic type; others are variants, e.g. ITQ works for short/quick interactions. All have a basic structure of: <Question>, <Interaction>, <Answer>, <Discussion>. Components are optional: <Question> <Answer> and <Question> <Discussion> are common combinations. All parts can contain many elements: paragraphs, lists, tables, etc.

<Interaction> creates interactive questions (single choice, multiple choice, matching, free text and more). These would be a pain to set up in Word so set up in oXygen and copy back into Word as *RawXML* (see [Escape to RawXML](rawxml)). Content can be tweaked once in Word, for example to correct a typo.

Activities are similar to boxes; they are visually and semantically enclosed. In Word, an activity is created between pairs of styled paragraphs, for example *ActivityHead* and *ActivityEnd* enclose an <Activity>. The question immediately follows *ActivityHead*; add a line styled *Answer* and/or *Discussion* to separate the question from answer and discussion. Any text on the *ActivityHead* line will be used as a heading; text on the *ActivityEnd*, *Answer* and *Discussion* lines will be lost. To create other activities, chose styles appropriately, for example *ExerciseHead*, *Discussion* and *ExerciseEnd*.

*Limitation*: Although SC allows it, I do not support nested boxes / activities in any permutation. Remember that all boxes allow *SubHeading* and *SubSubHeading*. If you feel, say, an activity needs a study note, maybe place it before/after?

Activity{resource1="activity"}{resource2="cloud\_based"}

This is the question.

Do you need a hint? Click the button if you do.

<RevealText> used to offer a hint

Answer

This is the answer.

Discussion

This is the discussion.

A paragraph of body text.

An exercise

This is the question.

More of the question.

Discussion

This is the discussion.

More discussion.

A paragraph of body text.

SAQ

This is the question.

Discussion

This is the discussion.

A paragraph of body text.

SAQ (with single-choice interaction)

Which three colours of the visible spectrum are camera sensors designed to capture?

Interaction

<SingleChoice><Wrong><Paragraph>Cyan, magenta and yellow</Paragraph></Wrong><Right><Paragraph>Green, blue and red</Paragraph></Right><Wrong><Paragraph>Red, green and yellow</Paragraph></Wrong><Wrong><Paragraph>Red, orange and blue</Paragraph></Wrong><Wrong><Paragraph>Yellow, red and blue</Paragraph></Wrong></SingleChoice>

Answer

Camera sensors are designed to capture red, green and blue in order to mimic the behaviour of the human eye.

You may find How is colour captured in a digital camera?olink/crossref required useful.

A paragraph of body text.

ITQ

This is the question.

Answer

This is the answer.

A paragraph of body text.

Reveal text: Can occur anywhere but can only be simple text, not paragraphs or other content.

A paragraph of body text.

See [Activities](https://learn3.open.ac.uk/mod/oucontent/view.php?id=185753) for the full list of activity types.

*Limitation*: SC supports <Multipart> for activities made of several parts, each of which is functionally a complete activity. (A motivation might be to avoid numbering parts individually.) I have not implemented this in Word: please use separate activities and combine in oXygen at handover where necessary.

# Images

You will want to paste images into a Word document so you can see what you are writing about but unfortunately these images will not transfer to XML. The VLE preview/publishing system requires images and other assets to be placed on a server. Following the recipe below will allow you to preview/publish images from a server such as [\\dog](file://dog/) (for drafts) or SharePoint (for final images), as well as see your images in context while writing.

Use the following styling on successive lines:

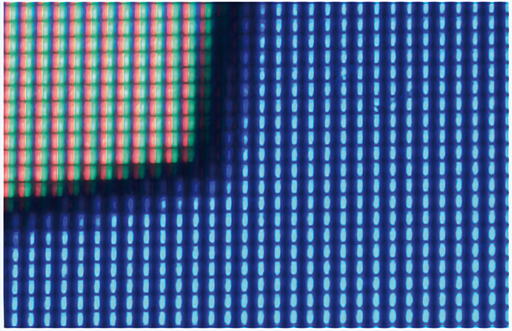
1. *Figure* for a paragraph in which the image is pasted (but note this image is a visual placeholder; the render will use the file at the URL given as *FigureSrc*).
2. *FigureSrc* for the URL of the image file on a web server (or pathname if on a file server)
3. *Caption* for text of caption (see earlier section for advice on figure [numbering](numbering))
4. *Alternative* for alt text if required
5. *SourceReference* to show where the figure originated (LDS don’t do this; they extract from rights database, insert at end of each document and keep themselves busy checking numbering)
6. *Description* for long description (one or more lines)

*Figure* must be used to generate the <Figure> tag; others are optional but should appear in order.

<*IMHO*> A good caption and figure description are more useful to more people than alt-text.</*IMHO*>

<*hobbyhorse*>The *SourceReference* can be used as acknowledgement for each figure, but I suggest that during authoring, you also use it as a cross-reference to the OU Rights Database aka ‘The Portal’. For every copyright image, get its unique ID and incorporate it into *SourceReference*. Also use *SourceReference* to note figures/images that do not need copyright clearance: OU drawn, screen dumps, Creative Commons, etc. Then you can run my custom transform in oXygen to generate an asset list for the current document should your curriculum manager or LDS project manager demand it; this report shows <SourceReference> and therefore essential rights info. To keep busywork to a minimum, it might be better to initially only inform LDS about images that definitely require copyright clearance. Those that don’t require clearance can wait to be added to the rights portal at final handover. This avoids trying to keep portal entries (e.g. figure numbers!) in sync with documents that are still being drafted. </*hobbyhorse*>

An image follows this paragraph:



https://openuniv.sharepoint.com/sites/tmodules/tg089/lmimages/tg089\_wk05\_f002.tif.jpg

Magnified screen showing individual pixels of an icon

© The Open University

Another image, this time with description.



https://openuniv.sharepoint.com/sites/tmodules/tg089/lmimages/tg089\_wk05\_f019.tif.jpg

The bright rose draws our eye

Courtesy of Margaret Salisbury

Two figures in elaborate masks and costumes during the Venice Carnival. One figure has a white costume, the other black, and both silhouetted against an evening sky. The figure in black offers a bright red rose to the figure in white.

Additional paragraph of description if required.

Another paragraph of description if required.

Back to a normal paragraph.

Another image



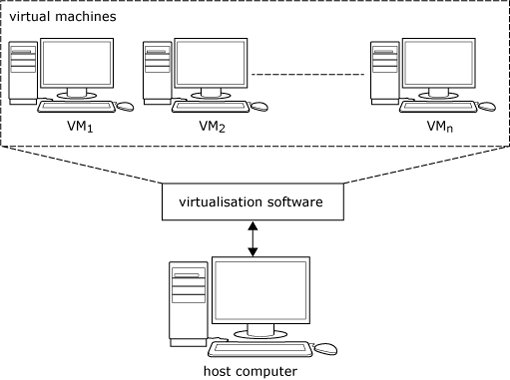
https://openuniv.sharepoint.com/sites/tmodules/tg089/session1e1/t189s01\_f05-03.jpg

Misty trees

TG089 course team, The Open University

A square format landscape showing two birch trees in silhouette against a misty background. The photo overall is low-key and has as a purplish tint.

Back to a normal paragraph.



\\dog\units\_non\_course\TG089\jpr2\tm129\_blk03\_tm129\_lin\_w1\_f01.tif.png

An image from \\dog

Back to a normal paragraph.

Image below to follow



https://openuniv.sharepoint.com/sites/tmodules/tm129/lmimages/missing.png

An image with no url given

Steve Granger, The Open University

A close-up of a rock rose with white petals and yellow centre. The centre of the flower is on the right rule of thirds. Two more blooms on the left top and bottom rule of thirds are out of focus in the background.

Images without a source url will show a default image which needs replacing as appropriate. The placeholder is hardwired as https://openuniv.sharepoint.com/sites/tmodules/tm129/lmimages/missing.png which works for now...



https://openuniv.sharepoint.com/sites/tmodules/tm129/lmimages/missing.png

An image with a bad url wouldn’t show as ‘missing’ but would generate an error when rendering

Steve Granger, The Open University

A close-up of a rock rose with white petals and yellow centre. The centre of the flower is on the right rule of thirds. Two more blooms on the left top and bottom rule of thirds are out of focus in the background.

Back to a normal paragraph.

<*hobbyhorse*> The issue noted earlier with regard to fuzzy equations on high resolution displays may also currently be apparent as fuzzy text in images, for example in hand-drawn figures and screen dumps. Worth a discussion with LDS staff to see if there has been any progress in this area…</*hobbyhorse*>

<*hobbyhorse*> And don’t get me started on how LDS processes result in poor image quality, lost colour spaces… Although I believe the handling of large images has now been improved – thanks, Tammy!</*hobbyhorse*>

Caption appearance

The intended appearance of a figure caption on the VLE is for the number to be bold but the caption itself to not be bold. This is controlled by the SC <Caption> and <Number> tags; you should not apply bold directly. I suggest doing no manual styling or numbering and using the numbering script.

# Media content

Some examples of rich media are given here and an explanation follow.

An introductory video.

Watch the following video to discover what the focus of Week 5 will be.

<MediaContent src="https://openuniv.sharepoint.com/sites/tmodules/tg089/lmaudio/tg089\_2022c\_vsc019\_1920x1080.mp4" height="1080" width="1920" type="video"><Caption>Video: Introduction</Caption><Transcript><Remark>[Text on screen: Enjoying colour]</Remark><Remark>[Text on screen: Dr Jon Rosewell, Senior Lecturer in Information &amp; Comms Technologies]</Remark><Speaker>JON ROSEWELL</Speaker><Remark>Welcome back. This week, we’re going to be looking at understanding and using colour. We take colour for granted. We see it wherever we look. </Remark><Remark>But if the essence of photography is light, then the essence of light is colour. So we really should pay attention to colour as we create images. And understanding some of the basic science of colour and how our brains make sense of colour in images will help you create better photographs. </Remark><Remark>This week begins with an exploration of the mechanisms of sensing colour in the eye and in the camera. The week then turns to considering how you might exploit colour in your photography. We’ll also give you a language with which to discuss colour as you talk about your own and others’ images. </Remark><Remark>The starting point for this week is to ask, how can the technology of digital cameras and displays begin to reproduce all the subtle colours that we see in nature? How many colours do you think your screen actually produces? If you look closely, you might be surprised. I hope you enjoy the week. </Remark><Remark>[MUSIC PLAYING] </Remark></Transcript><Figure><Image src="\\stcn-file-live\assets\poster-framer\99ea52863cc2fc78aa182a9e12feb29755e682d0.jpg"/></Figure></MediaContent>

An HTML5 interactive from TG089.

Choose a cone type by clicking on one of the three boxes: S cone (short), M cone (medium) and L cone (long). Then, drag the slider underneath the spectrum to illuminate the eye with different colours/wavelengths of light. Notice how the response of the cone varies with colour. Try each cone in turn.

<MediaContent src="https://openuniv.sharepoint.com/sites/tmodules/tg089/lmimages/tg089s05\_f2.2\_01.zip" type="html5" height="345" id="wk5\_act4" width="\*"><Description><Paragraph>An interactive figure to illustrate the responses of cones to different wavelengths. A slider at the bottom right below a spectrum can be moved to set the wavelength. To the top left are bars representing how strongly each cone type responds at that wavelength. Buttons below allow you to see the response of an S cone (blue), M cone (green) and L cone (red). At the top right is a graph plotting the relative response (on a scale 0 to 1) of a cone type against wavelength (on a scale 400 to 700 nm which aligns with the spectrum below). This graph shows three traces, each with a similar shape which rises to a peak and then falls. The response of an S cone (blue) peaks at about 450 nm, an M cone (green) peaks at about 560 nm, and an L cone (red) at about 580 nm. The L cone (red) response is somewhat broader than that of the other cones. </Paragraph><Paragraph>As the wavelength is changed, the bar showing the strength of response of a cone type rises and falls and the corresponding response curve is traced out on the graph.</Paragraph></Description></MediaContent>

Although each cone type responds strongly to one colour of the spectrum, they also respond, to a greater or lesser extent, to a range of colours. This means that light of any particular colour striking the eye will give rise to a combined response by the different types of cone.

Below is a poll not in an activity.

<MediaContent width="\*" type="html5" height="0" id="tm129\_b3\_t1\_poll02" src="https://openuniv.sharepoint.com/sites/tmodules/tm129/lmimages/simple\_poll.zip"><Parameters><Parameter name="options\_count" value="5"/><Parameter name="save\_mode" value="false"/><Parameter name="question" value="Do you expect to use Linux in the future (after TM129)?"/><Parameter name="option0" value="No, probably not"/><Parameter name="option1" value="Possibly – for personal use, study or hobby"/><Parameter name="option2" value="Possibly – for work or similar purposes"/><Parameter name="option3" value="Very likely – for personal use, study or hobby"/><Parameter name="option4" value="Very likely – for work or similar purposes"/></Parameters></MediaContent>

A poll inside an activity.

Activity 1: Poll

10 mins

Answer the question

<MediaContent id="sp-motivation-1-2-2b" type="html5" src="https://openuniv.sharepoint.com/sites/tmodules/tg089/lmdocs/simple\_poll.zip" width="\*" height="0"><Parameters><Parameter name="options\_count" value="3"/><Parameter name="save\_mode" value="false"/><Parameter name="question" value="I'm doing this course because I would most like to:"/><Parameter name="option0" value="Understand how to use my camera"/><Parameter name="option1" value="Learn how to manipulate images"/><Parameter name="option2" value="Improve my photography"/></Parameters></MediaContent>

And this one please:

<MediaContent id="sp-motivation-1-2-2c" type="html5" src="https://openuniv.sharepoint.com/sites/tmodules/tg089/lmdocs/simple\_poll.zip" width="\*" height="0"><Parameters><Parameter name="options\_count" value="3"/><Parameter name="save\_mode" value="false"/><Parameter name="question" value="I'm doing this course because I would most like to:"/><Parameter name="option0" value="Understand how to use my camera"/><Parameter name="option1" value="Learn how to manipulate images"/><Parameter name="option2" value="Improve my photography"/></Parameters></MediaContent>

Discussion

Most cameras, including smartphone cameras, have a selection of different exposure modes ranging from fully automatic to fully manual, with semi-automatic modes in between.

An HTML5 interactive from TM129.

You can choose any directory in an existing file system as a mount point over which another disk’s contents will be placed. The path to a file in a mounted file system is the merged path to the mount point and the path from the original (now hidden) root of the mounted disk to the file. In the figure below a USB stick has been mounted over the directory /mnt/usb.

<MediaContent type="html5" webthumbnail="false" src="https://openuniv.sharepoint.com/sites/tmodules/tm129/lmimages/tm129\_blk03\_fs\_files\_f08.zip" id="figure9" width="\*" height="275"><Caption>Mounting a USB stick onto a Linux system at <ComputerCode>/mnt/usb</ComputerCode></Caption><Description><Paragraph>An animated diagram to illustrate disk mounting.</Paragraph><Paragraph>Before the mount there are two separate file hierarchies: one for the Linux system and the other for a USB stick.</Paragraph><Paragraph>The Linux system has a file system starting at root / with directories bin, lib, mnt, and usr. The mnt directory has two subdirectories, usb and dvd, which are empty.</Paragraph><Paragraph>The USB stick has a file system starting at root / which has two directories, TM111 and TM129, and TM129 contains two files tma01.docx and tma02.docx.</Paragraph><Paragraph>When the animation is run, the entire tree of directories and files on the USB stick is shown sliding over to merge with the Linux system, the root of the USB system joining to the Linux system at the /mnt/usb directory.</Paragraph><Paragraph>After the mount there is therefore a single merged hierarchy on the Linux system. This is unchanged except that the /mnt/usb directory now has directories TM111 and TM129, and TM129 contains two files tma01.docx and tma02.docx.</Paragraph></Description></MediaContent>

The file tma01.docx on the USB stick has the path /mnt/usb/TM129/tma01.docx once mounted.

## Escape to RawXML{id="rawxml"}

An escape mechanism is used to preserve data structures as RawXML; this can be used to handle assets such as video, audio, interactives, ITQ interactions, quiz questions... For example this is an interactive asset held in <MediaContent>:

<MediaContent id="sp-motivation-1-2-2a" type="html5" src="https://openuniv.sharepoint.com/sites/tmodules/tg089/lmdocs/simple\_poll.zip" width="\*" height="0"><Parameters><Parameter name="options\_count" value="3"/><Parameter name="save\_mode" value="false"/><Parameter name="question" value="I'm doing this course because I would most like to:"/><Parameter name="option0" value="Understand how to use my camera"/><Parameter name="option1" value="Learn how to manipulate images"/><Parameter name="option2" value="Improve my photography"/></Parameters></MediaContent>

These complex structures have little or no text content and are best set up and tested in oXygen. Switch to oXygen Text view, copy the <MediaContent> or other element, paste into Word as text and style as *RawXML*; then leave it well alone. In future, this XML will be copied verbatim during conversion, recreating the tag that accesses media.

If you are reusing existing material with interactive assets, then the XML source will already exist and can be simply copied. For early stages of drafting, the asset may not yet exist: you might choose to use an image and description as a placeholder. At a later stage of production, LDS staff may share drafts using VLE pages and you can then incorporate the relevant XML into your draft.

# Test of heading levels (level 1){id="test-headings"}

Word Heading styles are converted to Structured Content elements that reflect the major hierarchical structure of a document as follows. A session/section/subsection ends where the next section begins; unlike a box, the end of a section is not marked explicitly.

Heading level and SC document structure

|  |  |  |  |
| --- | --- | --- | --- |
| **Word style** | **Structured Content element** | **Page navigation?** | **Hierarchical structure?** |
| Heading 1 | Session | yes | level 1 |
| Heading 2 | Section | yes | level 2 |
| Heading 3 | SubSection | yes | level 3 |
| Heading 4 | SubSubSection | no | level 4 |
| SubHeading or Heading 5 | SubHeading | no | no |
| SubSubHeading or Heading 6 | SubSubHeading | no | no |

Word styles *Heading 1* to *Heading 4* are converted to structural, enclosing SC tags. They must be used in strict hierarchy: you must not jump from *Heading 1* to *Heading 3*.

*Heading 1* to *Heading 3* will convert to <Session>, <Section> and <SubSection> which are reflected in the VLE navigation menu (so keep section titles short!) and in the contents page of a book. On the VLE, each of these start a new web page with Next / Previous navigation.

*Heading 4* <SubSubSection> does not start a new page; its heading and contents will appear under the enclosing <SubSection>. Once you are within a <SubSubSection>, it is also possible to use <SubHeading> and <SubSubHeading> (Word styles *SubHeading* and *SubSubHeading*). These are not structural and you can scatter them with abandon. If used, Word styles *Heading 5* and *Heading 6* will translate to <SubHeading> and <SubSubHeading>.

You can escape from these strict rules by means of <InternalSection>. This can be placed at any hierarchical level; it functions as an invisible [box](boxes) within a page and doesn’t cause a page break. It can have a <Heading> which is visually similar to the level 3 heading <SubSection><Title>. Within an internal section, you can freely place <SubHeading> and <SubSubHeading>. You can therefore use <InternalSection> to create manually organised pages which can be longer than the choppy style enforced by the standard level structure. (Short pages and clicking ‘Next’ was considered preferable to clicking on scroll bars, but that was before the world got used to swiping.)

OU house style (enforced by LDS) is outline numbering for levels 1–3, but not level 4. Internal sections and headings should escape numbering. See [Numbering](numbering) for advice.

## A Section (level 2)

Word style *Heading 2* becomes a <Section> which starts a new page and is in navigation menu.

A paragraph of body text.

Can’t put headings in a Section

A paragraph of body text.

A paragraph of body text.

### A SubSection (level 3)

Word style *Heading 3* becomes a <SubSection> which starts a new page and is in navigation menu.

A paragraph of body text.

Can’t put headings in a SubSection

A paragraph of body text.

A paragraph of body text.

#### SubSubSection (level 4)

Word style *Heading 4* becomes a <SubSubSection> which *doesn’t* start a new page and *isn’t* in navigation menu.

(Confusion over terminology! By the time it gets to structured content, a SubSubSection has a Heading, not a Title; I guess logic is that titles are only given to things that start pages and are part of site navigation. Well, it confused me...)

Word *Heading 1-4* are converted to structural, enclosing SC tags. You can also use *SubHeading* and *SubSubHeading* which can be placed anywhere in a SubSubSection and which are wholly unstructured. Their appearance should be sensible relative to Headings 1-4. Word *Heading 5* and *Heading 6* also translate to <SubHeading> and <SubSubHeading>.

Subheading in sub sub section

A paragraph of body text.

A paragraph of body text.

SubSubHeading in sub sub section

A paragraph of body text.

A paragraph of body text.

## Another Section (level 2) with internal sections

<InternalSection> can be used anywhere. It acts like a box, ie it is a containing structure, has a heading and can contain (unstructured) <SubHeadings> and <SubSubHeadings>. It is probably most useful as an alternative to a level 3 heading; an alternative that doesn’t cause a break to a new page.

For example, if a Heading 3 style were used here, a new page would be generated leaving very little on this page.

Internal section heading

A paragraph of body text.

A paragraph of body text.

Subheading in internal section

A paragraph of body text.

A paragraph of body text.

SubSubHeading in internal section

A paragraph of body text.

A paragraph of body text.

This paragraph is actually outside the <InternalSection> although that isn’t obvious once rendered. Is it detectable in print? No idea.

Another internal section

More than one <InternalSection> on a page is allowed.

By the way, a level 4 <SubSubSection> would also not split the page, but you can’t place a level 4 <SubSubSection> directly into a level 2 <Section>; you would need an intervening level 3 <Section> as container and that would split the page anyway. You pays your money and has no choice...

Internal sections aren’t in the navigation menu, but <CrossRef> can jump to an <InternalSection>.

## Yet another Section (level 2)

### A SubSection (level 3)

#### Down to SubSubSection (level 4)

Although the rule is that level 1, 2 and 3 always start a new page, where there is no content before the next heading (for example, no introductory paragraphs), the titles are rendered together to avoid having pages with a title but no content. The navigation panel will contain all titles for levels 1-3.

# Using the converter

## Environment

Software can be found on GitHub <https://github.com/JonRosewell/struct-auth>.

You will need to use oXygen. oXygen should have IT/LDS customisations already set up. If not, follow: [OXYGEN-OSEP: Installation](https://learn3.open.ac.uk/mod/oucontent/view.php?id=185735).

To replace the IT/LDS conversion with mine:

1. Use Windows File Explorer to look at this directory: C:\Users\[youroucu]\AppData\Roaming\com.oxygenxml.author\extensions\v26.0\frameworks\structuredcontent\structuredcontent\xslt
2. Locate the file xhtml2sc.xsl.
3. Rename it to xhtml2sc.bak.
4. Copy in my version of xhtml2sc.xsl.
5. Start oXygen.

## Converting Word to SC

Briefly:

1. write a styled Word document
2. copy the content to clipboard
3. paste into a skeleton document in oXygen
4. click Render to preview
5. click the Save link to publish.

Conversion from Word to structured content is done on the fly when pasting Word into an OU Structured Content XML document. This means fragments can be converted, or the full text of a document can be pasted into an empty skeleton file.

The easiest way of obtaining a skeleton file is to download XML from an existing published page (ie visit a VLE page for your module, click the Downloads icon at bottom of page, and choose XML format). Open this file in oXygen. Delete all <Sessions> to leave a nearly empty <Unit> into which you will paste new content. Save this skeleton e.g. as empty.xml.

Alternatively you can create a new SC document in oXygen using File > New > Framework Templates > Structured Content > Body Matter Skeleton. You will need to fill in some of the <Item> attributes and initial elements as appropriate for your module.

Now switch to your source document which should be open in Word. Select and copy all content (Ctrl+A, Ctrl+C). Switch back to oXygen, open the empty.xml template and immediately save under a new name to preserve your template. Place your cursor inside <Unit> and paste (Ctrl+V) the content copied from Word. Check for any validation errors.

Use the oXygen Structured Content > Render menu or toolbar button to see a preview. Check for any further warnings.

At this stage you can publish your document to a VLE site by clicking the Save link on the preview. The default website and location is controlled by the <CourseCode> and the <Item> id attribute. You can usefully publish to a module team workspace for colleagues to review your current draft, including any <AuthorComment> and <EditorComment>; colleagues can make their own comments using track changes on your Word version.

Rinse and repeat.

### File conversion

The suggested way of working is as above: copy/paste Word content into a skeleton XML file within oXygen. This requires oXygen with OU IT/LDS customisation; internally, this converts Word content into HTML and then XHTML before my conversion (replacing the official version) translates into structured content.

An alternative workflow in outline would be:

1. Save your Word document as HTML.
2. Convert HTML to XHTML (XHTML 1.0 Transitional).
3. Convert XHMTL to SC XML using html-to-sc.xsl (which calls the normal xhtml2sc.xsl translator but also provides a skeleton container).
4. Upload and preview at [learn2 upload page](https://learn2.open.ac.uk/mod/oucontent/upload.php).

In principle, this could be done without oXygen, although it would require alternative tool support: an XSLT processor and an HTML-to-XHTML converter. oXygen provides both of these tools, and it is possible to use the above workflow using oXygen *without* the OU IT/LDS customisation. Other toolchains could be used; I have not investigated any but for instance Eclipse would be a good bet since oXygen is based on it.

## Converting SC to Word

Before you start work on existing VLE documents, note that there are several options:

* Obtain the XML master and edit in oXygen. This is the quickest and safest option for small changes.
* The normal ‘official’ process is to start with a Word version and use track changes; after handover, an editor will make corresponding changes to the master XML. Unfortunately this is slow and error prone, so not recommended.
* **Recommended** for all non-trivial changes and definitely for major rewrites: (down) convert the XML into styled Word and use my (up) converter.

I can offer a (semi-)automated down conversion from an existing XML master document to styled Word. This preserves all content, including rich content such as images, video and interactions, and styles the Word document so that it can immediately be previewed and published. You can then edit the Word document, taking advantage of quick preview and even publish the final document.

*Limitation*: The automated down conversion turns XML to styled HTML which can be opened and edited in Word. However, there are additional steps required, firstly to set up oXygen and then, for every file converted, to embed external images into Word .docx. This is a pain, so I am happy to do this myself for occasional files on request. If you expect to do this for many documents, [set up the script](setupscripts) for sc-to-html.xsl and run it against the master XML document. Open the resulting HTML file in Word and [embed the image files](embedimages).

*Limitation*: Although having both down- and up-conversions between Word and XML provides the potential for round-trip working, I can’t claim this would make sense as a routine way of working. There are a few limitations on what can be written in Word (for example table borders are not currently preserved) and additional effort is required to convert into .docx. But it is worth doing when it is time for a rewrite of existing material.

## Setting up scripts{id="setupscripts"}

The conversion from Word to SC XML automatically runs the appropriate script thanks to the OU IT/LDS customisation of oXygen; no further setup is essential.

However, you can also configure oXygen to run additional scripts: the IT/LDS customisation provides a few scripts and I can offer a few more, for example for auto-numbering. These scripts are accessed through the oXygen Transformation Scenarios view. You may first want to set up an oXygen project for your work on a module.

The next section shows how to set up the script for auto-numbering; the same procedure can be used for other scripts.

### Set up the autonumber script

1. Open the Transformation Scenarios view by clicking on its tab or using menu Window > Show view > Transformation scenarios.
2. Click the Add button in the view’s toolbar and choose XML transformation with XSLT.
3. Complete the settings shown in the table below, making use of dropdowns provided.
4. When you wish to autonumber a file, open it, check Autonumber (and no others) in the Transformation Scenarios view and click on Apply.
5. Preview and save the result.

|  |  |  |
| --- | --- | --- |
|  | **Setting** | **Value** |
| **Overall** | Name: | Autonumber |
| **XSLT tab** | XML URL: | ${currentFileURL} |
|  | XSL URL: | file:/C:/Users/[…]/number-sc.xsl |
| **Output tab** | Save as: | (blank) |
|  | Open in editor: | (checked) |
|  | Show in results view as: | (none checked) |

The main toolbar also has buttons for configuring and running transformations which you may find more convenient. The transformation set up should only be required once. The oXygen environment will usually remember which transformation is used with which file.

### Available scripts

Other scripts are available:

* number-sc.xsl: auto-numbering
* sc-to-html.xsl: SC XML to styled HTML which can be imported into Word
* html-to-sc.xsl: converts XHTML *files* originating from styled Word into SC XML (not normally required)
* asset-list.xquery: produces an asset list for a document (set up as XML transformation with XQuery)

These scripts need to be set up as above, changing details as appropriate.

## Embedding images in Word{id="embedimages"}

This advice mainly applies to down-conversion from SC to Word via HTML – ignore otherwise. Info here is *not required* for conversion from Word to SC XML.

Word will open and save HTML documents. HTML images are external files whose location is in the *src* attribute of <img> elements. Word preserves this external linkage, even if the document is subsequently saved as a .docx format file, but this can cause permission/access issues when moving/sharing documents. To avoid this, you can embed images in .docx files.

To do this, open an HTML document in Word and:

1. Open the Links dialogue – type this in the Word search bar because it is not shown on toolbar.
2. Select all images listed
3. Check Save picture in document and click Update now
4. Again select all images and click Break link
5. Save the document in .docx format.

**Note**: The Links dialogue should close automatically when all external images have been dealt with. If it shows persistently, it suggests an image cannot be found. Use Change source to remedy, choosing a placeholder image if necessary.

**Note**: If original images are on OU SharePoint, Word may refuse to link / display them. A work around is as follows:

1. Instead of opening the HTML file in Word, open it first in a browser.
2. Check that images are displayed.
3. Save a local copy of the complete web page – this creates a local folder with copies of images and rewrites the <img> *src* attributes to match.
4. Open this saved document in Word and proceed to embed images as above.

(It seems OU Word treats OU SharePoint as a security risk but I cannot fix trust settings to allow this; if anyone knows how, please share...)

**Note**: Another reason that images may not display is due to LDS image workflow. SC files may refer to the .tif or .eps original and rely on the publishing backend to automatically generate a .jpg version (of possibly dubious quality!). Edit the XML master, adding .jpg (sometimes .png) extension (e.g. a123\_fig01.eps.jpg) to make things work.

**Note**: Another reason that images may not display is a *src* file path which references DavWWWRoot. Edit the XML master to change following this example:

src="\\openuniv.sharepoint.com@SSL\DavWWWRoot\sites\tmodules\tg089\lmimages\t189s06\_f07-02.jpg"

src="https://openuniv.sharepoint.com/sites/tmodules/tg089/lmimages/t189s06\_f07-02.jpg"

(Any similar *src\_uri* attributes can be left, edited or deleted.)