

XSL

XSL is a language designed to add formatting to XML documents, which it does in a process called a transform. This process is referred to as XSLT. Here are a couple of examples of how it works in practice:

- Using an XSLT processor (a piece of software which can apply an XSL document to an XML document) you open an original XML document. In the same processor, you can open an XSL document in a second window. You click on a button – ‘Transform’ – and the XSL rules are applied to the XML. In a third window appears the result document.
- Using InDesign, you `Import XML`. Upon import, a dialogue box asks you if you would like to `Apply XSL`. You browse to the location of the XSL file and hit `Import`. As the XML is imported into the InDesign document, InDesign acts as an XSLT processor and uses the rules defined by the XSL to format the XML.

We use XSL in this way to transform ONIX data to produce many documents, from catalogues and AIs to flyers and web pages.

Whilst you can use InDesign as an XSL processor, it cannot handle the most recent update to the XSL language (XSLT 2.0). In these instances, the XML needs to be transformed before it can be imported into InDesign. Using a standalone processor is also useful when you are amending your XSL. For this we use one of the free XSLT processors available on the net.

A simple, free application is Cooktop (<http://www.xmlcooktop.com/>) The website looks a bit ropey but it's an excellent application for rapidly processing and checking your XSL. However, it is no longer being developed and does not support XSLT 2.0.

A robust and comprehensive, and also free, alternative is Eclipse (<http://www.eclipse.org/>) combined with the Orangevolt XSLT 2.0 plug in (<http://eclipsexslt.sourceforge.net/chapter-N10356.html>).

You can also buy XSLT processors. Altova Spy and oXygen XML editor are two market leaders. Whilst you have to pay, both companies offer free 30 day trials.

XSL: THE LANGUAGE

Like XML, XSL is written in English. For the purposes of using it to transform ONIX XML, there are only a handful of commands, which you can learn easily. It's particularly easy to start with a piece of finished XSL and tweak it by, for instance, altering the order of the commands.

The output of an XSLT is effectively another XML file which you can import into

InDesign. InDesign is very particular about the order in which XML elements appear. The order of the elements in the incoming XML file must match the order in which they appear in the document. XSL allows you to take bits of the ONIX standard which are relevant to you and re-order them.

Here is an example of some XSL.

```
<authorbio><xsl:value-of select="ContributorStatement"/></authorbio>
```

Whilst it looks a little technical at first, let's break it down.

You'll notice that there are three parts to this line, each in a set of tags: <>. The first and last part mean that in the output document an element will be created called <authorbio>. The middle part defines what that element will be. It does this by saying 'Hey, I'm an XSL command, so use the rules of XSL to interpret me. Select the value of whatever's at the element Contributor Statement in the XML file. and put it between these two tags, <authorbio> and </authorbio>'

This next bit of XSL can be interpreted the same way.

```
<price><xsl:value-of select="SupplyDetail/Price/PriceAmount"/></price>
```

'This is an XSL command. Find out what the value of this bit of the ONIX record is and select it, then put it between these two tags.'

The XSL commands you need to know are:

XSL: `value-of` (see above)

XSL: `choose` XSL: `when` (allows you to set parameters for the selection of values)

XSL: `for-each` (allows you to apply a set of commands for each product in the XML file)

XSL: `sort` (allows you to sort the output file).

And the principle you need to remember is that everything has an opening and a closing tag.

That's about it, aside from some standard header information. There's loads more you can learn, but those commands will get you by. Here's a sample full XSL file for you to see the commands in action.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<BK xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xsl:version="1.0">
<xsl:for-each select="ONIXMessage/Product">
<xsl:sort select="PublicationDate" lang="en-gb"/>
<xsl:sort select="Subject[SubjectSchemeIdentifier = '23']/SubjectCode" lang="en-gb"/>
<xsl:sort select="Contributor/KeyNames" lang="en-gb"/>
<Book>
    <xsl:variable name="bk" select="Title/TitleText"/>
    <xsl:variable name="xx" select="ProductFormDetail"/>
    <xsl:variable name="dt" select="PublicationDate"/>
    <xsl:variable name="isbn" select="ProductIdentifier[ProductIDType = '03']/IDValue"/>
    <xsl:variable name="bic" select="Subject[SubjectSchemeIdentifier = '23']/SubjectCode"/>
    <xsl:variable name="rights" select="SalesRights/RightsTerritory"/>
<title><xsl:value-of select="$bk"/></title>
<authorfirstname><xsl:value-of select="Contributor/NamesBeforeKey"/></authorfirstname>
<authorlastname><xsl:value-of select="Contributor/KeyNames"/></authorlastname>
<sellingpoints><xsl:value-of select="OtherText[TextTypeCode = '27']/Text"/></sellingpoints>
<description><xsl:value-of select="OtherText[TextTypeCode = '01']/Text"/></description>
<authorbio><xsl:value-of select="ContributorStatement"/></authorbio>
<price><xsl:value-of select="SupplyDetail/Price/PriceAmount"/></price>
<format>
<xsl:choose>
    <xsl:when test="$xx = 'B104'">A Format Pbk</xsl:when>
    <xsl:when test="$xx = 'B105'">B Format Pbk</xsl:when>
    <xsl:when test="$xx = 'B106'">Trade pbk</xsl:when>
    <xsl:when test="$xx = 'B304'">Colour pbk</xsl:when>
    <xsl:when test="$xx = 'B501'">Jacketed HB</xsl:when>
    <xsl:when test="$xx = 'B502'">Slipcased HB</xsl:when>
    <xsl:when test="$xx = 'B221'">Colour pbk</xsl:when>
    <xsl:when test="$xx = 'B306'">Library HB</xsl:when>
    <xsl:otherwise>Unknown binding</xsl:otherwise>
</xsl:choose></format>
<pages><xsl:value-of select="NumberOfPages"/> pages</pages>
<productid><xsl:value-of select="ProductIdentifier[ProductIDType = '03']/IDValue"/></productid>
</Book>
</xsl:for-each>
</BK>

```