EDITEUR

ONIX for Books

Product Information Message

Application Note: Validating ONIX 3.0 files

PC edition

One of the major advantages of using XML is that you can use a process called 'validation' to check that your data file is structured correctly for your application. But validation isn't always a familiar process, and it's remarkable how many ONIX adopters do not routinely use validation as a way of checking their ONIX data, or use only the most basic validation tools that don't check as comprehensively as they could.

The most basic question to ask about an ONIX file – or any XML file – is, 'is it well-formed?' *Well-formed* means that it obeys the rules of XML itself. That is, is the data enclosed by XML tags? Are the tags properly matched, with each start tag like <tag> matched with an end tag like </tag>? Are they properly nested, tags one inside the other? And does the file avoid using the & and < characters in the data? (Using them in the markup is okay.) If your XML file obeys all the syntax rules of XML, it's well-formed.

But well-formed isn't enough. The next step is to ensure your file is 'valid'. That is, does it match the ONIX specification? For example, are all the tags actual ONIX tags? Are they in the right order for ONIX? And have you included all the mandatory tags? A valid ONIX file must be well-formed, but it must *also* meet these extra requirements.

The simplest possible validation

There are three ways that an ONIX file can be validated using tools provided by EDItEUR. The most basic is via the ONIX DTD (Document Type Definition) [1]. It checks the tags, but nothing more. Validation using the 'classic' XSD is more powerful, as it checks the tags *and* some of the data [2]. The 'strict' XSD is the most comprehensive test, as it performs all the classic checks and adds hundreds of further tests on the data [3] [4].

What does the DTD check? It checks all the tags are real ONIX tags – and remember, the tags are case-sensitive, so if you're using long or reference ONIX tags, <Product> is okay but <PRODUCT> or <product> are not. It checks the opening tags match the closing tags (so every <Product> must match a </Product> tag. It checks the nesting, so all the <Product> tags must be inside an <ONIXMessage> tag. It checks the tag order, ensuring for example that <Subject> comes before <Audience>. And it checks *cardinality* – some tags are mandatory, some are optional, some can be repeated and some cannot. However, it doesn't check that the publication date is a real date – in fact, you can put pretty much anything inside each of the tags and it will still appear valid.

So how can you validate an ONIX file using the DTD? You need four things – the ONIX file itself, an XML parser, a text editor and the ONIX DTD.

---

[1] The DTD defines all the tags you can use in an ONIX file – <Contributor>, <PriceAmount> and so on. There are about 450 tags in all. It also defines the order they have to be used in, how each tag can be nested inside others, whether tags are mandatory or optional, whether they are repeatable or not. To some extent, the DTD defines what ONIX *is*

[2] The XSD does all the things the DTD does, and in addition defines what kind of data each tag can contain – an integer, a decimal, some text, or codes from a codelist

[3] The extra business rules applied by the strict XSD spot errors in identifier check digits, non-existent dates, internal inconsistencies like e-books with physical measurements, and a host of others

[4] This of course means that an invalid ONIX file can *appear* to be valid if you just check it with the DTD, because the DTD isn't comprehensive. But at least the DTD is a good place to begin

In this Application Note, we'll use a free-of-charge application called XML Notepad [5] as the parser. If you Google 'XML Notepad 2007' the Microsoft download will work, but it's an old version. The most up to date version of the standalone installer can be found at the Github URL several entries further down the Google results [6]. For the text editor there are a number of free options: three reliable and easily-found standbys are Atom, Notepad2 or Notepad++. Just make sure you're not using a word processor. And you need the DTD itself, which you can get from the EDItEUR website [7]. Download the Zip file of DTDs, unzip and store the DTD files somewhere convenient on your computer (there are several files that together make up the DTD). For this Application note, the DTD files are stored in a directory called 'DTD' within a directory 'ONIX_3.0.7' off your main user home directory (that's normally 'C:\Users\Yourname' on Windows 10). This isn't necessarily the *best* place, but it keeps the path names short. Your file paths may differ.

Finally, XML Notepad is an excellent tool for XSD schemas but it doesn't support DTD validation as an option. Now there's no real need to do DTD validation once you're doing XSD schema validation, as XSD validation is more powerful, but we'll demonstrate DTD validation using a command line XML parser. XMLStarlet supports most major operating systems, including a Windows 32-bit version:

```
http://xmlstar.sourceforge.net/overview.php
```

Download the 'xmlstarlet-1.6.1-win32.zip' file (or a later version) and unzip it. Once installed, the program runs from Windows' Command Prompt (if you're unfamiliar with it, type 'cmd' into the Search box on the Taskbar) and it's easier still if you copy the XML Starlet's executable file 'xml.exe' into the ONIX_3.0.7 folder.

Now you need to prepare your ONIX file. It is a good idea to try all this with a file you know is valid before you tackle any real-world ONIX. You can download a valid sample file from the EDItEUR website [8] and put it in the ONIX_3.0.7 directory too.

ONIX files may need to be prepared with a statement that helps a parser find the right files. Most ONIX files, using reference tags, begin like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<ONIXMessage xmlns="http://ns.editeur.org/onix/3.0/reference" release="3.0">
```

or with short tags:

```
<?xml version="1.0" encoding="UTF-8"?>
<ONIXmessage xmlns="http://ns.editeur.org/onix/3.0/short" release="3.0">
```

Open the sample ONIX file using your text editor. In a real-world file, the *xmlns* attribute might be missing, and of course your file might be using short tags like <ONIXmessage> (note the lower case M) instead of long (reference) tags that begin with <ONIXMessage>. Add the *xmlns* attribute if it isn't there already, and remember you may need to adjust 'reference' to 'short' as necessary through the rest of this Application note.

Using a command line parser

In this example we've used the long tag ONIX sample file called 'Onix3sample_refnames.xml' in the directory 'C:\Users\Yourname\ONIX_3.0.7'. Open the Command Prompt window, and navigate to the folder in which you put the XML Starlet executable, the DTD and the sample files. Enter:

```
cd ONIX_3.0.7
dir
```

---

[5] EDItEUR has no affiliation with the developer of XML Notepad 2007, nor with any developers of other XML tools like XML Spy

[6] https://github.com/microsoft/XmlNotepad

[7] https://www.editeur.org/93/Release-3.0-Downloads/#Schema%20defs

[8] https://www.editeur.org/files/ONIX%203/ONIX_Books_3.0_sample_5.zip The rest of this Application note assumes you're using this sample file

Since you're in the directory ONIX_3.0.7, you should see the DTD folder, the ONIX example files and the xml.exe executable file all listed. Test XML Starlet by entering '`xml -h`' at the command prompt – this should return a menu of basic commands. If all is as it should be, try typing this:

```
xml val -e -d DTD\ONIX_BookProduct_3.0_reference.dtd Onix3sample_refnames.xml
2> dtd_1.txt↵
```

Note that's all one line before pressing Return (↵) at the end. The -d parameter makes the 'xml val' command check the sample file against the requirements in the DTD. The part in red should be the location of the DTD, and the part in green should be the sample ONIX file. The response appears on the screen as:

```
Onix3sample_refnames.xml - valid
```

and the output file 'dtd_1.txt' is blank because there are no errors. If you've made a mistake with the command, then there will probably be no response on the screen, but the output file is still created.

But with real-world ONIX you might just as often get an error. To demonstrate this, make a *copy* of the valid file and call it Onix3sample_refnames copy.xml'. Open the copy in your text editor and modify it using cut and paste. Move the last <Subject> composite (it's about half way down the file) so it's *after* the <Audience> composite, and save.

Now re-run the command to validate this modified copy. Use the name of the edited file instead of the green filename (since the edited filename includes a space, surround the whole thing with quotes), and use 'dtd_2.txt' as the output file name. This is what the result might look like on screen:

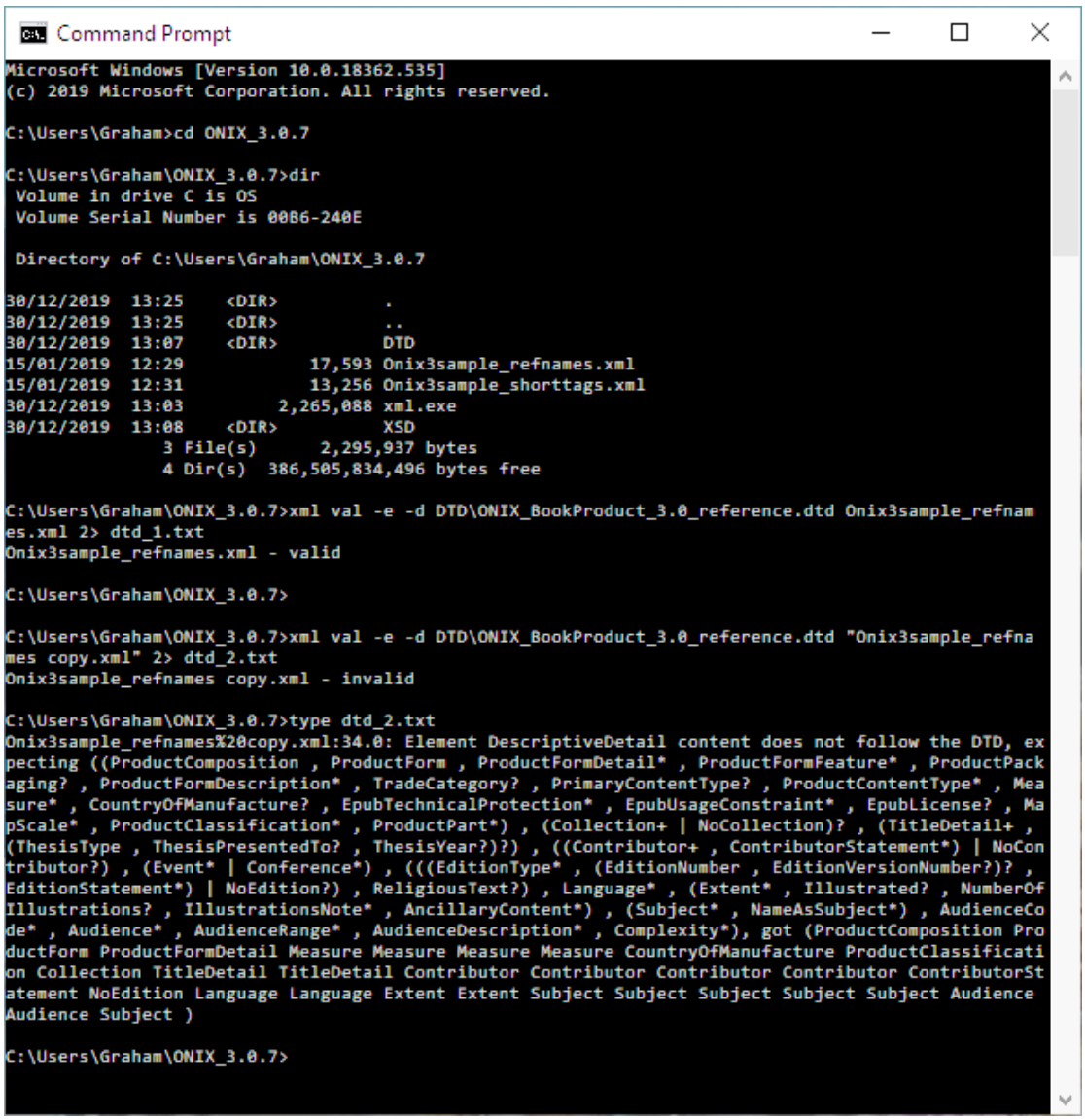
```
Onix3sample_refnames copy.xml - invalid
```

The output file contains an error message, and to read it you can enter:

```
type dtd_2.txt
```

The full error message starts with the filename and line number, and should look something like this:

```
Onix3sample_refnames copy.xml:34.0: Element DescriptiveDetail content does not
follow the DTD, expecting ((ProductComposition , ProductForm ,
ProductFormDetail* , ProductFormFeature* , ProductPackaging? ,
ProductFormDescription* , TradeCategory? , PrimaryContentType? ,
ProductContentType* , Measure* , CountryOfManufacture? ,
EpubTechnicalProtection* , EpubUsageConstraint* , EpubLicense? , MapScale* ,
ProductClassification* , ProductPart*) , (Collection+ | NoCollection)? ,
(TitleDetail+ , (ThesisType , ThesisPresentedTo? , ThesisYear?)?) ,
((Contributor+ , ContributorStatement*) | NoContributor?) , (Event* |
Conference*) , (((EditionType* , (EditionNumber , EditionVersionNumber?)? ,
EditionStatement*) | NoEdition?) , ReligiousText?) , Language* , (Extent* ,
Illustrated? , NumberOfIllustrations? , IllustrationsNote* , AncillaryContent*)
, (Subject* , NameAsSubject*) , AudienceCode* , Audience* , AudienceRange* ,
AudienceDescription* , Complexity*), got (ProductComposition ProductForm
ProductFormDetail Measure Measure Measure Measure CountryOfManufacture
ProductClassification Collection TitleDetail TitleDetail Contributor
Contributor Contributor Contributor ContributorStatement NoEdition Language
Language Extent Extent Subject Subject Subject Subject Subject Audience
Audience Subject )
```



You'll immediately see that 'helpful error messages' aren't a feature of XML validation. The error is not on line 34 at all, but much further down the ONIX file. But you might notice that it says "Expecting …, got …", and the error message ends with '…Subject Audience Audience Subject)', when the order should be Extent, Subject, Audience.

Validating with the XSD schema

Using a DTD is a very basic type of validation, and it only catches errors with tagging. What if you've made an error with a code from a codelist? The DTD won't catch it. The XSD schema is a more advanced tool, and *can* spot this type of error. So ensure you download the XSD files [9], unzip and save them in a folder 'XSD' inside ONIX_3.0.7.

You can validate using the XSD at the command prompt with XML Starlet just as before:

```
xml val -e -s XSD\ONIX_BookProduct_3.0_reference.xsd Onix3sample_refnames.xml
2> xsd_1.txt↵
```

Note the -s parameter instead of the -d. This should be valid, and will leave behind an empty xsd_1.txt file.

[9] again, from https://www.editeur.org/93/Release-3.0-Downloads/#Schema%20defs

And the file with the error?

```
xml val -e -s XSD\ONIX_BookProduct_3.0_reference.xsd "Onix3sample_refnames
copy.xml" 2> xsd_2.txt↵
type xsd_2.txt↵
```

This time, the error report is much more specific:

```
Onix3sample_refnames copy.xml:210.12: Element '{http://ns.editeur.org/onix/
3.0/reference}Subject': This element is not expected. Expected is one of (
{http://ns.editeur.org/onix/3.0/reference}Audience,
{http://ns.editeur.org/onix/3.0/reference}AudienceRange,
{http://ns.editeur.org/onix/3.0/reference}AudienceDescription,
{http://ns.editeur.org/onix/3.0/reference}Complexity ).
```

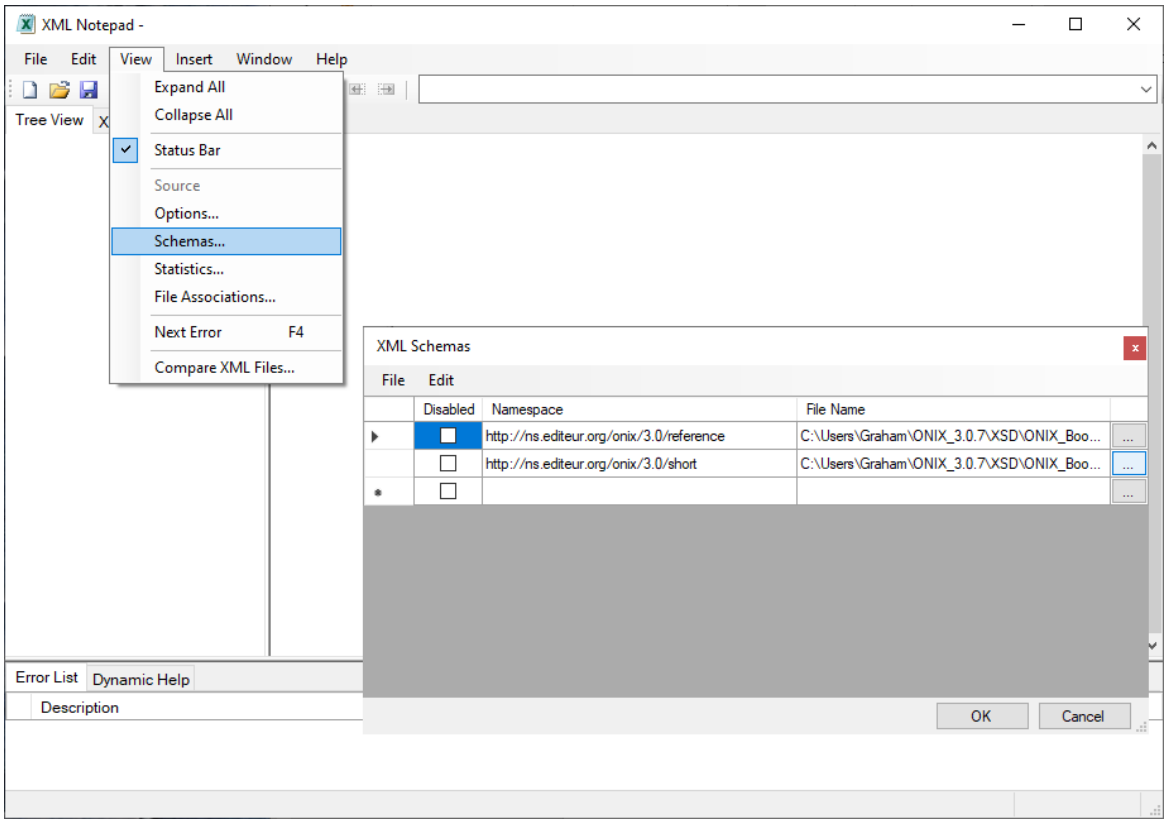
The error messages from XSD validation are generally a little less arcane than if you use the DTD. You'll see here the error is at line 210 – which is fairly close -- and it says 'This element is not expected' (this is the same error as before, where <Subject> occurs after <Audience>).

Now, using the 'classic' XSD checks for many more issues than the DTD. In particular, it checks that codes from codelists are real codes. If not, you could get an error like this:

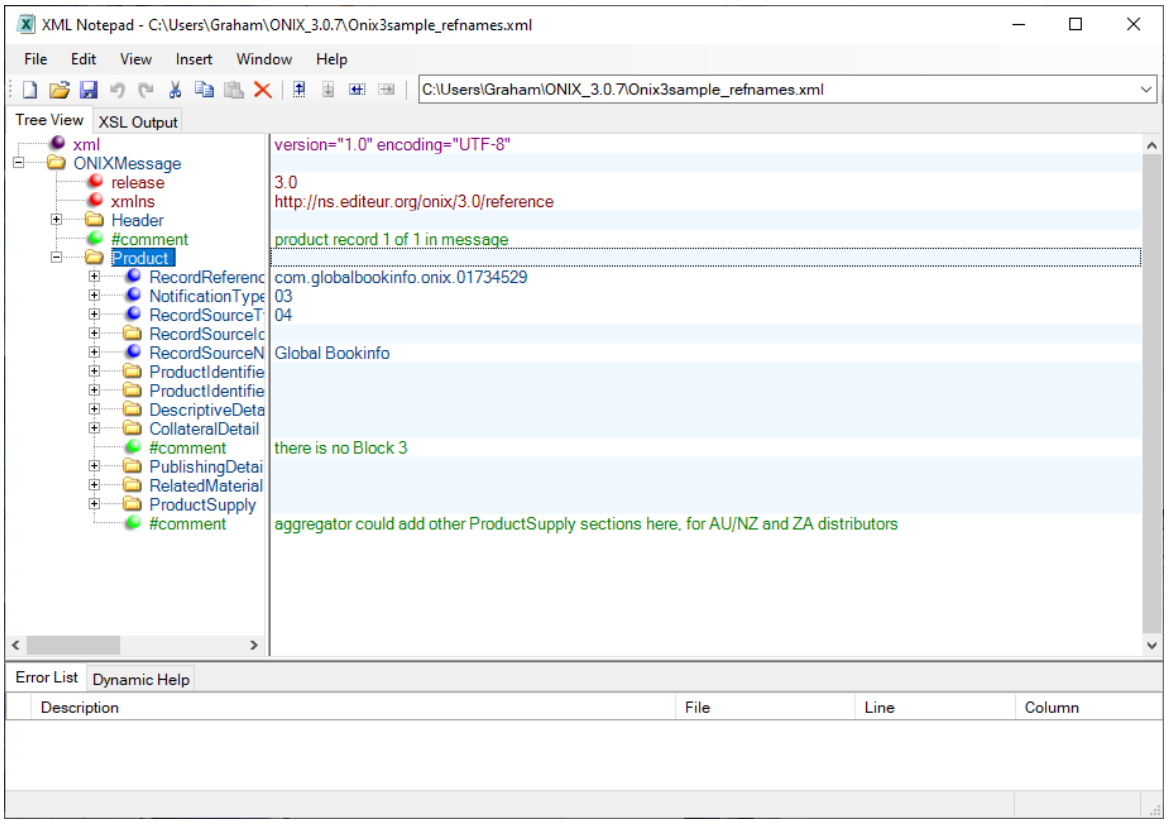
```
Element '{http://ns.editeur.org/onix/3.0/reference}ProductIDType': [facet
'enumeration'] The value '99' is not an element of the set {'01', '02', '03',
'04', '05', '06', '13', '14', '15', '17', '22', '23', '24', '25', '26', '27',
'28', '29', '30', '31', '35'}.
```

XML Notepad provides a convenient graphical user interface for schema validation, and it can be used to edit the ONIX file too. Start the application (type 'XML notepad' into the Taskbar Search box).

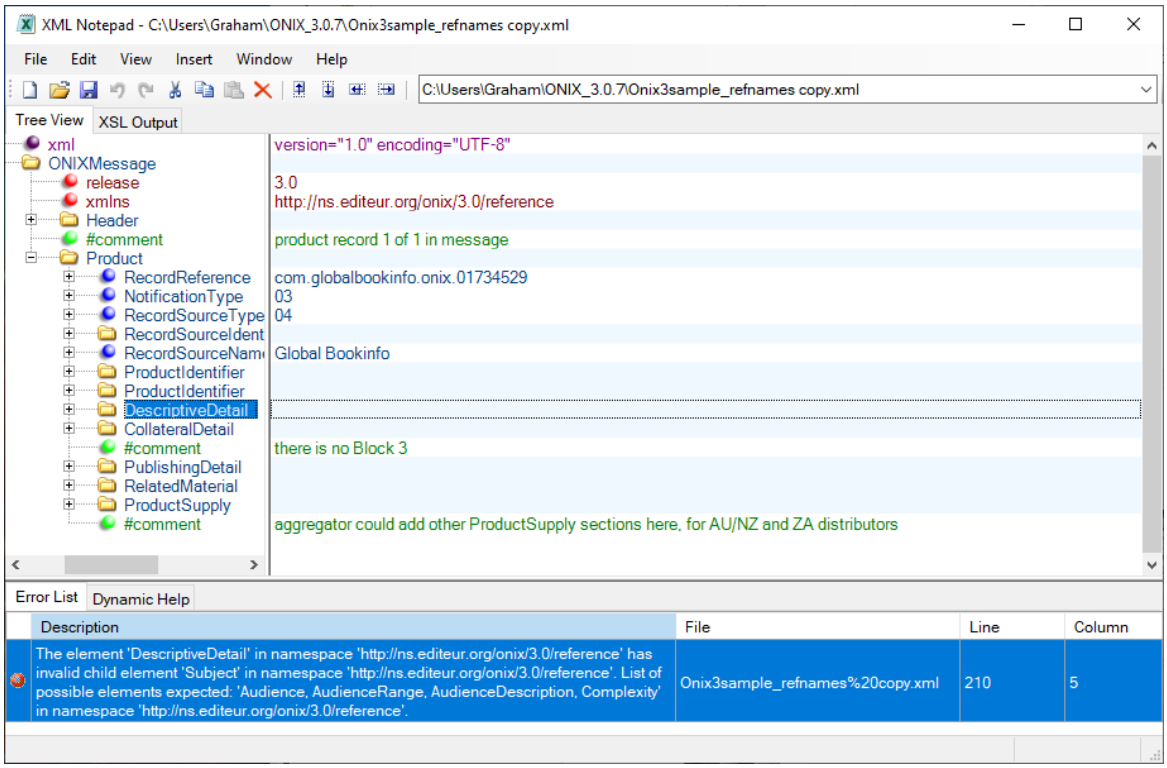
The application requires a one-time setup to add the ONIX schema paths to the software. Choose Schemas from the View menu and use the […] button to browse to the ONIX_3.0.7\XSD directory and add the reference and short .xsd files. Once the two XSD schemas are listed in the XML Schemas window, click OK, and that's it.



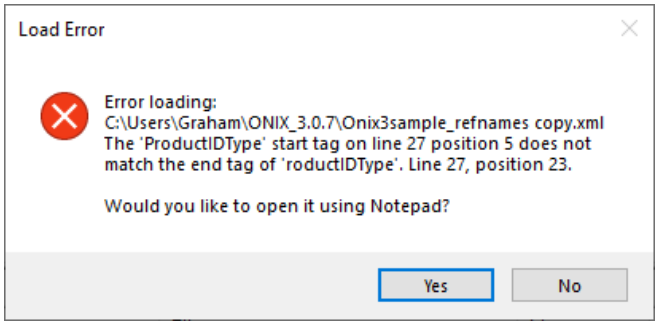
First, try the valid ONIX file by choosing Open from the File menu and browsing to ONIX_3.0.7/Onix3sample_refnames.xml. Hit OK and the file should open. If the error list is empty, then the file is valid.



So now use File > Open to load the Onix3sample_refnames copy.xml file (which contains the error). XML Notepad immediately shows you the error. One of the benefits of using XML is that the error messages are often a little easier to interpret than that provided by XML Starlet:



Occasionally, you may get an ONIX file that will not load, with the following error message:



Wth XML Notepad, an XML file needs to be at least 'well formed' before it can be loaded – start tags matched by end tags, and tags nested correctly. It may be best to avoid editing the file to fix the tags in Windows Notepad (the default choice) – click No and use your favourite text editor. Fix the tags on the line number indicated, save, and re-load in XML Notepad.

The view of the ONIX file provided within XML Notepad may appear unconventional – quite different from the view provided by a text editor or web browser. The 'tree view' shows the XML 'nodes' (the individual ONIX tags) on the left, and you can expand or collapse composites with the [+] buttons. The data content of each node is listed on the right, and you can edit the ONIX by editing the contents, or by adding new nodes with a right-click on the tree.

You could fix the Onix3sample_refnames copy.xml file by moving the out-of-place <Subject> tag. Open <DescriptiveDetail> in the node list, then click and drag the last <Subject> node upwards a little so it's with the other <Subject> tags – and the error message should go away.

But while the XSD is more powerful than the DTD, it can still check only the tagging, the codelists, and a few of the data elements. It can ensure that <EditionNumber> is actually a number, but it cannot highlight a non-existent date, require a *textformat* attribute, or verify the check digit on an ISBN. The 'strict' XSD can…

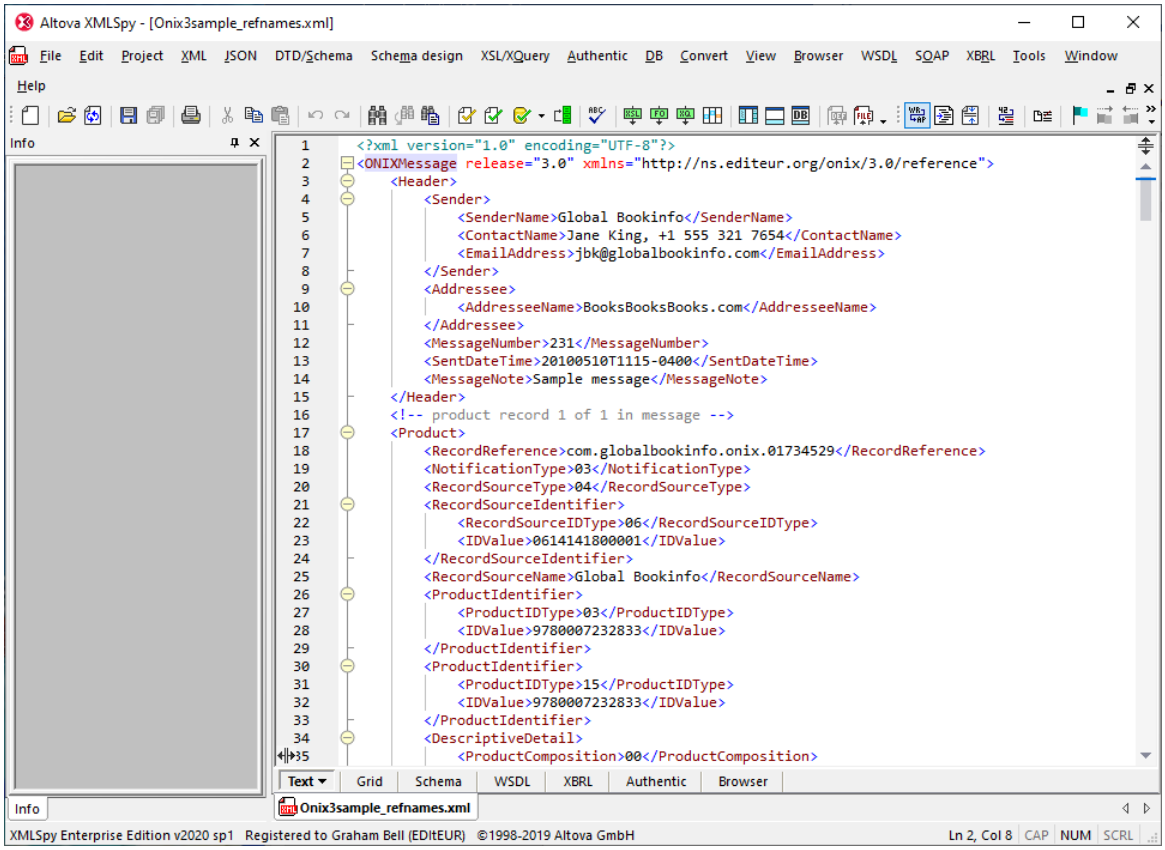
Validating with the strict XSD schema

XML Notepad 2007 and the XML Starlet command line parser work with the ONIX DTD and with the 'classic' XSD. They do *not* work with the 'strict' XSD.

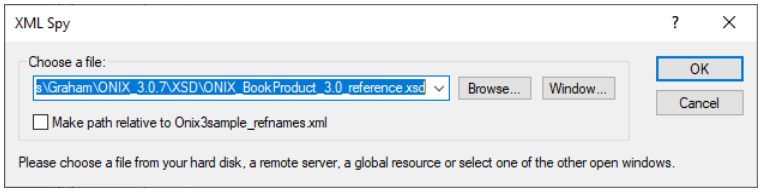
This strict XSD is the most advanced way of checking the validity of an ONIX 3.0 file, but it only works with recent versions of three specific XML parsers: Saxon, Xerces and Raptor. Here we'll use XML Spy (powered by Raptor). A free trial version of XML Spy is available. Saxon and Xerces are included in oXygen XML Editor (available for both Windows and Mac from https://www.oxygenxml.com), which it is featured in the Mac version of this Application note.

The strict XSD can validate a wide selection of the data in an ONIX file and ensure a high degree of internal consistency. But be aware that it's not infallible: the ONIX *Specification* encompasses a range of business rules and not all are explicitly checked even by the strict schema.

To validate the sample file using XMLSpy, open it and it should look something like this:

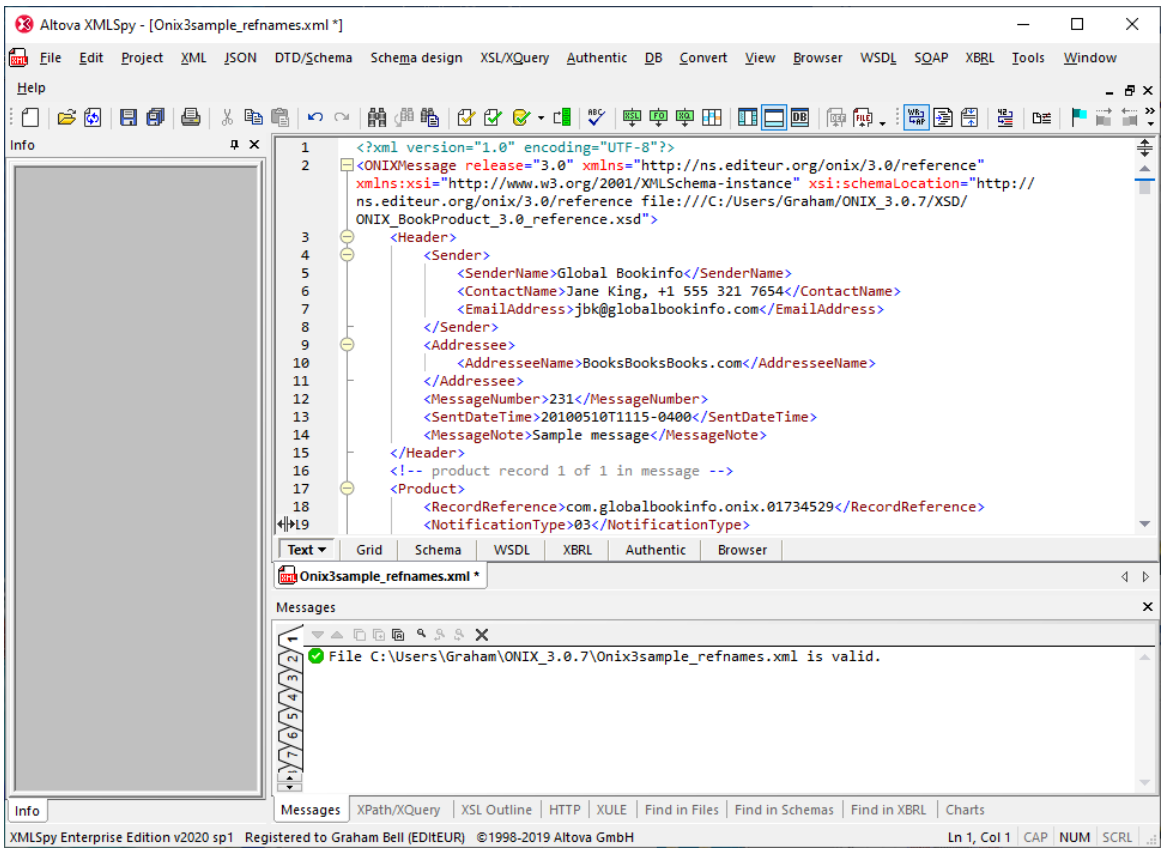


The ‘Validate’ button is the white dog-eared rectangle with the green tick, top centre of the window. However, you need to tell XML Spy in advance which XSD file to use [10], so choose DTD/Schema > Assign schema… from the menu bar. The dialog box should look something like this:

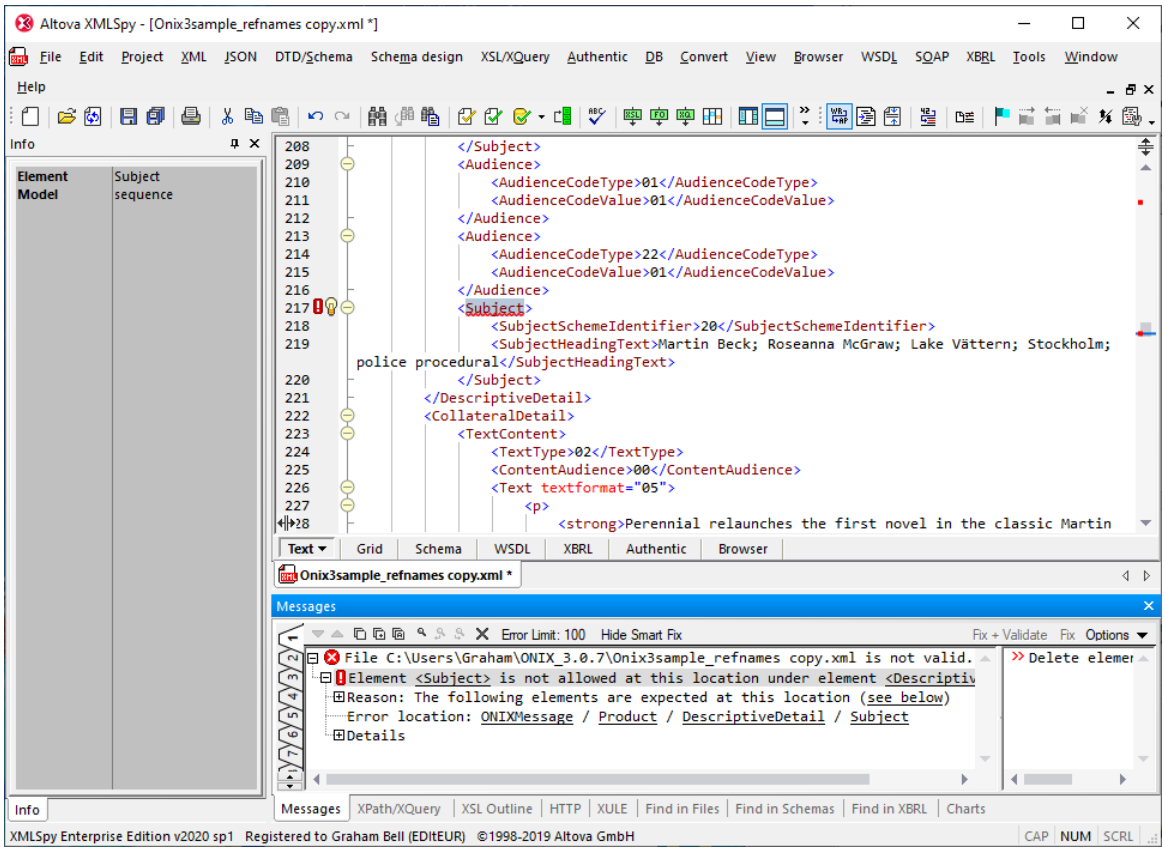


In this case, you can see the ‘classic’ XSD schema is selected. The strict XSD has ‘_strict’ on the end of the filename. Click on OK, and XML Spy will validate your ONIX file and present the results. You’ll see that XML Spy adds some extra attributes to the <ONIXMessage> tag. There should be no errors in the Messages pane:

[10] XML Spy can use DTD, XSD or strict XSD



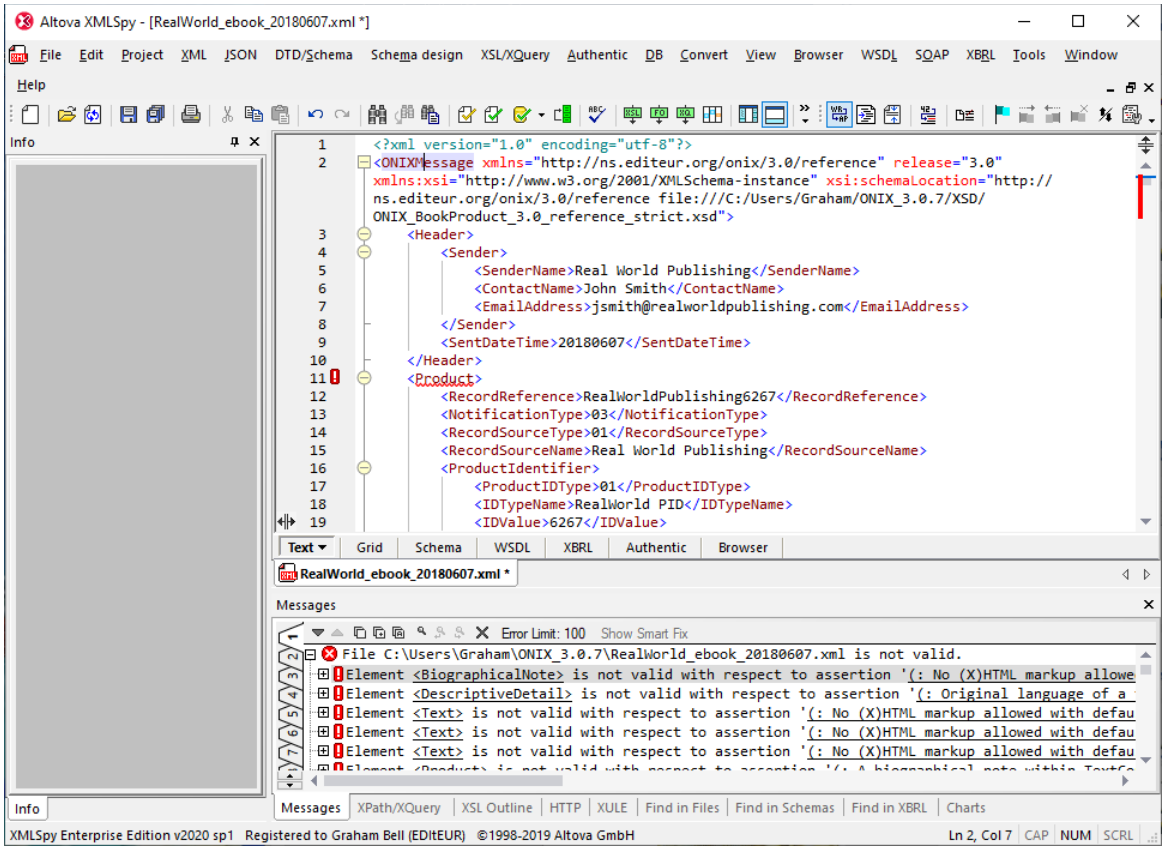
Now do the same thing with the edited copy of the sample file (which contains the '<Audience> before <Subject>' error). You'll see the error listed at the bottom of the window:



With XML Spy you don't need a separate text editor – it allows you to edit the ONIX file directly in the window, so it's easy to fix the error then re-validate by clicking the Validate button. You'll find that XML Spy shows you errors as you edit, and you can run a full validation just by clicking on the Validate icon at any point. (Just remember to remove the extra *xmlns:xsi* and *xsi:schemaLocation* attributes if you're going to send the ONIX file to anyone else.)

So what about the strict XSD? Download the strict schema files [11] and put them in the same place as the classic XSD files – the two strict XSD files share the same codelist and XHTML subset files as the classic XSD. Use the DTD/schema menu > Assign schema… dialog to select the XSD with '_strict' at the end of the name, and validate. With any sizeable real-world file, you are very likely to see many errors – even if it validates correctly with the classic XSD.

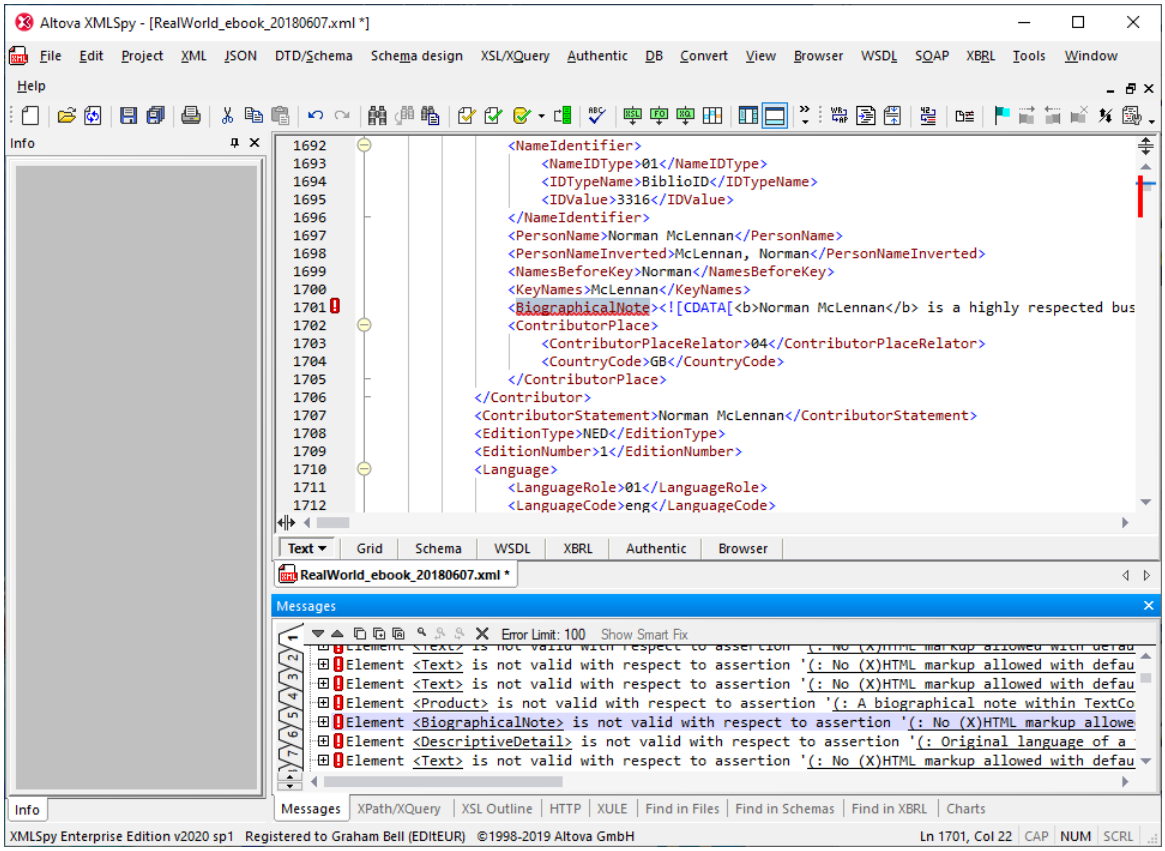
Here's an ONIX file from a real publisher containing 110 Product records (the names have been changed). It validates without errors using the classic XSD, but contains nearly 850 errors if validated with the strict XSD (XML Spy shows only the first 100):



You may notice that the errors produced by the strict XSD are a little more explanatory.

[11] from https://www.editeur.org/93/Release-3.0-Downloads/#Advanced%20schema%20defs

A great feature of XML Spy is that clicking on one of the errors shows *where* in the file it occurs:



Now you have validation working with the strict XSD and some sample files, it's time to tackle some real-world ONIX. Most real-world ONIX that's produced using an off-the-shelf product management system is valid when using the classic XSD, but you'll often find a surprising number of issues with the strict XSD. Just don't get spooked by the sheer number of errors, as you usually get the same error over and over again, once per record systematically, and for some types of error, feedback to the system vendor is the best way to get them fixed.

A final caveat

No validation of any kind will protect you if the data itself isn't correct! Strict validation will point out the error if you claim a publication date of Feb 30th, but a date of Feb 28th is 'valid' – even if the correct date is actually January 28th.

Graham Bell and Tom Richardson
EDItEUR / BookNet Canada
23rd November 2019

Notes

You never need to validate using the DTD *and* the XSD. The classic XSD includes all the checks the DTD does, and the strict XSD includes all the checks of the classic XSD. Just use the most advanced one you can.

You don't need to do this to follow the steps in this Application note, but **sometimes, if you want to use a DTD, you need to edit the file to add a DOCTYPE**. To do this, open the ONIX file in your text editor and add the extra line in red (your filename will be different):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ONIXMessage SYSTEM
"C:\Users\Graham\ONIX_3.0.7\ONIX_BookProduct_3.0_reference.dtd">
<ONIXMessage xmlns="http://ns.editeur.org/onix/3.0/reference" release="3.0">
```

and save the modified ONIX file. This extra DOCTYPE line is *only* needed for DTD validation, with *some* validation tools, and must *not* be included in any ONIX you send to another organisation.

ONIX 2.1 *must* include a DOCTYPE declaration, and you will need to edit the filename of the DTD to match your system. *As transmitted*, the DOCTYPE must use 'http://www.editeur.org/onix/2.1/reference/onix-international.dtd' or the short equivalent – which means you might have to edit it *again* before sending the validated ONIX on to someone else. (With ONIX 2.1, if you use XSD validation without the DOCTYPE, it can often mark a perfectly valid file as invalid, because the XSD alone does not recognise the use of named character entities like … or –.

XML Spy does not appear to be able to use the embedded Schematron rules in the strict schema (at least by default). Strict validation can provide warnings *even if your file is valid*, for example if the ONIX file uses any deprecated elements or deprecated values from codelists. It will also note whether your ONIX uses any features from more recent revisions (*eg* 'Requires at least some features of ONIX 3.0.6'). However, this is only available with the Saxon and Xerces parsers, for example within oXygen – see the Mac version of this Application note. XML Spy also cannot use the RNG schema (which is equivalent to the classic XSD), whereas oXygen can.

Validation of large files (say more than 500 Product records) can require a lot of memory in your machine, so until you are familiar with the process, stick to relatively small files.