ONIX for Books

Summary of changes for ONIX for Books 3.0 revision 1

Revision 1 of ONIX 3.0 (known as 3.0.1) is a minor upgrade to the ONIX 3.0 format: most of the changes increase the flexibility of the format for use with East Asian languages (particularly Japanese and Chinese), and in multi-language supply chains, but other changes will potentially be useful for all ONIX users.

The outline of these changes was considered and approved by the ONIX International Steering Committee at its Frankfurt 2011 meeting, and detailed documentation has been circulated and commented upon by the ONIX National Groups around the world. The changes are detailed in an updated *Specification* and new DTD and schema files for ONIX 3.0.1, and an outline of the changes is included here. Additions have also been made to the *ONIX 3.0 Implementation and Best Practice Guide*. ONIX 3.0.1 is published alongside Codelists Issue 16, which is the earliest issue of the codelists that provides full support for the small number of new data elements introduced with 3.0.1.

None of the changes create any new mandatory elements or extra requirements for data providers. Consequently ONIX 3.0.1 is entirely backwards compatible with ONIX 3.0.

Note that item 13 below also has implications for ONIX 2.1.

Contents

1. Removal of ASCII character limitation in <Header>
2. Incorporation of glosses into text data elements
3. Incorporation of non-alphabetic sorting information
4. Addition of explicit sequence information for title elements
5. Addition of a title statement
6. Addition of a publication order within a collection
7. Addition of per-product contact information
8. Changes to cardinality of text data elements
9. Addition of new *textscript* attribute
10. Other minor changes to attributes
11. Deprecation of the <Reissue> composite
12. Deprecation of <AudienceCode>
13. Change to data type of <SubjectSchemeIdentifier>
14. Change of namespace for ONIX for Books Release 3.0
15. Naming and migration

1. Removal of ASCII character limitation in <Header>

In earlier versions of ONIX, there is a documented limitation that prevents the use of extended characters in any data elements within the ONIX message header. This appears to be an unnecessary legacy of ONIX 1.0. It rules out Cyrillic, Greek, Arabic, Chinese or Japanese characters in Header data elements, and even prevents the use of extended Latin characters which are necessary for most European languages. This limitation was removed in ONIX 2.1 rev.04 for Japanese use, and it is removed from ONIX 3 in similar fashion.

This limitation was not enforced in earlier versions of the ONIX schema, so the change affects the documentation only.

Note the change affects only data elements in the message header, as the elements within Product records have never been constrained to ASCII in this way. Note also that all ONIX syntax elements (the markup – element names and attribute names) remain ASCII-only, and numeric and date fields plus those fields that carry a codelist value are also limited to ASCII.

This change has no effect on current ONIX data, but recipients should be prepared to receive any XML-legal characters native to the overall language, script and character set used in the message in elements within the affected Header elements in future messages.

2. Incorporation of glosses into text data elements

Glosses, also called ruby text, are brief text annotations attached to particular words or phrases in the main text. They can be used to refine or explain the meaning (as in 'glossary') or pronunciation of a word or phrase of the main text, and are common in East Asian writing systems (for example, in Japanese and Chinese) to provide related phonetic information.

Facilities for such ruby glosses were added to ONIX 2.1 rev.04, and have been added to ONIX 3.0 in exactly the same fashion.

Method 1: Glosses in data elements that use XHTML or HTML markup

To incorporate a ruby gloss on text within in these elements, the established XHTML 1.1 markup for either 'simple' or 'complex' ruby should be used:

```
<Text textformat="05"><p>… <ruby><rb>村上春樹</rb><rp>(</rp><rt>むらかみ
はるき</rt><rp>)</rp></ruby> …</p></Text>
```

For simple ruby as shown, the whole of the annotated word or phrase is enclosed in a <ruby> element (in blue above). The base characters (Japanese Kanji in this case, highlighted in purple) are enclosed in an <rb> element, and the gloss (Japanese Hiragana characters in red) is enclosed within an <rt> element. Optionally, an <rp> element (in green above) can appear both before and after the <rt> element – its content should be displayed *only* if the rendering system cannot properly render the <rt> element as a gloss. This example should be displayed like this:

むらかみ はるき
… 村上春樹 …

or – if the system cannot support display of the gloss above the base text – like this:

… 村上春樹 (むらかみ はるき) …

HTML markup is the same, but of course it must be enclosed in <![CDATA[…]]> section delimiters. Note that such gloss markup in HTML 4 is not standardised, but works in most major browsers. The current HTML 5 draft uses a different markup that omits <rb> and <rp>, though this is subject to continuing discussion and HTML 5 may yet revert to using the <rb> tag.
For further details of ruby gloss markup, see Sections 1.2.1 and 1.2.2 in
http://www.w3.org/TR/ruby/.

Method 2: Glosses in data elements that *do not* include XHTML or HTML markup

To incorporate glosses into text data elements *without* XHTML or HTML markup, Unicode *interlinear annotation delimiters* should be used (see Section 3.6 in
http://www.unicode.org/reports/tr20/).

These characters can be inserted as numeric character references ￹ (at the beginning of the annotated phrase, in blue below), and &#fffb; (at the end of the annotated phrase, in

orange). The base characters (Simplified Chinese Hanzi in this case, in purple) are divided from the gloss (Chinese Pinyin, in red) with ￺ (in green below):

```
<TitleText>&#xfff9;残唐五代史演义传&#xfffa;Cán táng wǔdài shǐ yǎn yì
zhuàn&#xfffb;</TitleText>
```

This should be displayed as:

Cán táng wǔdài shǐ yǎn yì zhuàn
残唐五代史演义传

or:

残唐五代史演义传 (Cán táng wǔdài shǐ yǎn yì zhuàn)

Choice of which display method to use will depend on context and the technical capabilities of the recipient. Glosses are displayed above or following horizontal text, or to the left of text written vertically.

These Unicode interlinear annotation delimiter characters are for internal use within applications – in this case, 'within' ONIX – and cannot be rendered by web browsers or other display systems. They should be converted into the equivalent HTML/XHTML markup before display, for example if the data is used on a website:

```
&#xfff9;  =>  <ruby><rb>
&#xfffa;  =>  </rb><rt>  or  </rb><rp>(</rp><rt>
&#xfffb;  =>  </rt></ruby>  or  </rt><rp>)</rp></ruby>
```

Note that these interlinear annotation delimiters can be included 'natively', without using numerical character entities, if the text encoding allows. But practically, they may be simpler to include as numerical entities like '￻'.

The two methods for glosses are mutually exclusive within any single data element

Which method to use depends on whether XHTML markup is used for that particular data element. The XHTML method (Method 1) is *only* available for the following 24 data elements:

- <AncillaryContentDescription>
- <AudienceDescription>
- <BiographicalNote>
- <BookClubAdoption>
- <CitationNote>
- <CopiesSold>
- <ConferenceTheme>
- <ContributorDescription>
- <ContributorStatement>
- <FeatureNote>
- <IllustrationsNote>
- <InitialPrintRun>
- <MarketPublishingStatusNote>
- <PrizeJury>
- <PromotionCampaign>
- <PromotionContact>
- <PublishingStatusNote>
- <ReissueDescription>
- <ReligiousTextFeatureDescription>
- <ReprintDetail>
- <SalesRestrictionNote>
- <Text>
- <TitleStatement> (see 5. below)
- <WebsiteDescription>

and the use of XHTML tags within these elements must be signalled by including the *textformat* attribute with value 05 in the element start tag. HTML gloss markup can also be used with *textformat* 02 provided the data element containing HTML markup is enclosed within a CDATA section.

It is also possible to use the Unicode method (Method 2) in the 23 elements listed above, with *textformat* 06 (Default text format), provided the sender and recipient of the ONIX data agree and the character set allows characters ￹, ￺ and ￻. It is not possible to include glosses at all (using any method) with *textformat* 07 (Basic ASCII text).

For data elements other than the 23 listed above – for example in titles or contributor names – only the Unicode method may be used.

This extension obviously has no effect on current ONIX data, but provides a vital facility for ONIX users in China and Japan.

3. Incorporation of non-alphabetic sorting information

For some languages, the collation (sorting) order of titles, names is not based directly on the characters used in the title or name. A Japanese name rendered in Kanji is sorted phonetically, and there may be more than one pronunciation of a particular Kanji 'character'. Thus two names, rendered identically in written form but phonetically distinct, may sort differently. Phonetic information may be required, and may also be distinct from any gloss supplied with the text. Phonetic sorting may also be used in Chinese, as an alternative to sorting by the number of strokes in each character.

Phonetic information for collation purposes should be included in ONIX data elements using the *collationkey* attribute. This new attribute may be used on any element where a *language* attribute may be used (though the *language* attribute does not need to be present). For example:

```
<TitleText collationkey="エイカイワシリーズ"> … </TitleText>
```

In Japanese text, collation information would typically be provided using Hiragana. In Chinese text, collation information may be provided in Pinyin.

Collation keys may in principle be used on almost any textual data element, but are only of use on elements likely to be used for sorting multiple product records, including collection title and title elements, and contributor names. The following data elements can carry the *collationkey* attribute:

- <TitleText>
- <TitlePrefix>
- <TitleWithoutPrefix>
- <Subtitle>
- <PersonName>
- <PersonNameInverted>
- <TitlesBeforeNames>
- <NamesBeforeKey>
- <PrefixToKey>
- <KeyNames>
- <NamesAfterKey>
- <SuffixToKey>
- <LettersAfterNames>
- <TitlesAfterNames>
- <CorporateName>
- <CorporateNameInverted>

> Note – glosses are intended for display (and possibly for search), whereas *collationkey* attribute data is intended only for sorting. In many use cases, *both* have to be provided. In these cases, they should be provided separately, even though it is likely that they will often include the same phonetic information (…though they do not have to be identical.)

This change has no effect on existing ONIX 3.0 data.

4. Addition of explicit sequence information for title elements

For a complex publication title, various parts of the title that make up the complete product title are provided in multiple repeats of the <TitleElement> composite. For a product that has say, two elements, at collection and product level, the structure would typically look like this:

```
<TitleDetail>
        <TitleType>01</TitleType>
        <TitleElement>
                <TitleElementLevel>01</TitleElementLevel>
                …
        </TitleElement>
        <TitleElement>
                <TitleElementLevel>02</TitleElementLevel>
                …
        </TitleElement>
</TitleDetail>
```

For some products, the publisher prefers the collection title to precede the product-level title, and for others the publisher prefers the collection title to follow the product-level title. Originally in ONIX 3.0, <TitleElementLevel> indicated the relative importance of a title element (collection or product level), and *the order of the two <TitleElement> composites within the XML data* was intended to indicate the data supplier's preferred display order for the two elements – in the above example, this is product-level followed by collection-level.

However, in other areas of an ONIX message, the order of repeats of the same composite is not significant – <Contributor> for example has an optional <SequenceNumber> element to ensure the order of contributor names can be made explicit, and this is particularly significant where multiple roles are involved, or where contributors are split between P.5 and P.7 of the record.

An optional <SequenceNumber> data element identical to that used with <Contributor> is now available for use within <TitleElement>, to make explicit the preferred display order for the various elements of a title. For an example given in the ONIX 3.0 Specification, *A Game of Thrones: A Song of Ice and Fire, Book 1*, the proposed new structure would be:

```
<TitleDetail>
        <TitleType>01</TitleType>
        <TitleElement>
                <SequenceNumber>1</SequenceNumber>
                <TitleElementLevel>01</TitleElementLevel>
                <TitlePrefix>A</TitlePrefix>
                <TitleWithoutPrefix>Game of Thrones</TitleWithoutPrefix>
        </TitleElement>
        <TitleElement>
                <SequenceNumber>2</SequenceNumber>
                <TitleElementLevel>02</TitleElementLevel>
                <TitlePrefix>A</TitlePrefix>
                <TitleWithoutPrefix>Song of Ice and Fire</TitleWithoutPrefix>
        </TitleElement>
        <TitleElement>
                <SequenceNumber>3</SequenceNumber>
                <TitleElementLevel>01</TitleElementLevel>
                <PartNumber>Book 1</PartNumber>
        </TitleElement>
</TitleDetail>
```

(Note the first element is at product level, the second at collection level and the third at product level again. However, the three title elements could be supplied in any order within the XML without altering the meaning.)

It must be understood that not all ONIX data recipients would immediately be able to respect the preferred display order as indicated by sequence number, just as they do not necessarily reflect the preferred order as indicated by the order of elements in the XML – retailer's web pages often have fixed layouts that dictate the placement of each element. However, this addition is intended to discourage practices such as carrying collection information as a subtitle in order to force a retailer to display the collection information in a subsidiary position.

This change would have no effect on existing ONIX data.

5. Addition of a title statement

The list of contributors in a Product record is backed up with an optional <ContributorStatement> element, giving a concatenated list of contributor names and roles. This is a simple string, of little use for sophisticated searching of the metadata, but potentially of use to recipients for display purposes, particularly where a naive concatenation of the individual contributor names and roles would not be suitable. A similar, optional

<TitleStatement> has been added to carry the complete collection and main title, in the order and with any interposed punctuation that the publisher prefers. This should be used for display purposes *only*. Note that provision of an unstructured title statement as the sole title metadata is not acceptable – the structured title is still required.

This change has no effect on existing ONIX data.

6. Addition of a publication order within a collection

Titles forming a collection are occasionally published 'out of order' – that is, the order of publication is not necessarily the logical reading or narrative order, or the numbered order of the complete collection. For example, number 3 in a pre-planned five-part collection may be published first, followed by 1, 2, 4 then 5.

In ONIX for Books 2.1 rev.04, the data element <PubSequenceNumberWithinSeries> is used. In ONIX 3.0, this has been generalised: a new, optional and repeatable composite <CollectionSequence> can carry information about the product's order within a collection. This new composite sits within <Collection> in P.5. It may be provided even if all collection title elements are in P.6, and may be provided for either publisher or ascribed collections.

The structure of the new composite is as follows:
```
<CollectionSequence>
      <CollectionSequenceType>02</CollectionSequenceType>
      <CollectionSequenceNumber>3</CollectionSequenceNumber>
</CollectionSequence >
```

The <CollectionSequenceType> element and the new associated Codelist 197 allow multiple 'types' of sequential order (including the publication order) to be specified – for example original publication order for collected works that were first published outside of any collection, or an order associated with an adaptation that does not follow the sequential order of the collection itself. The values proposed for this codelist in Issue 16 are:
    01 Proprietary
    02 Title order (confirmation of the order specified by volume or part number)
    03 Publication order of products within the collection
    04 Temporal/narrative order within products in the collection
    05 Original publication order (for works first published outside this collection)
Value 01 requires use of the <CollectionSequenceTypeName> element to provided a short explanatory label for the particular sequence.

The <CollectionSequence> composite may be repeated to deliver different ordinal positions in different types of sequence for the same product within the collection. For example, for the well-known *Chronicles of Narnia* children's novels by C.S. Lewis (*The Lion, the Witch and the Wardrobe* and six others), the original publication order and the narrative order within the collection are quite different.

This change has no effect on existing ONIX data, as provision of a sequential order is entirely optional. Note where the products in a collection are numbered (*Volume 6*, *Part 3* and so on) the numbered order is provided within <TitleElement> but may be confirmed using collection sequence type code 02.

7. Provision of per-product contact information

There is a common requirement to communicate the name of a contact attached to an individual product, in addition to any technical contact names that can be included in the Header. Any contact named within <Sender> in the header is responsible for the overall

message data and is often in practice a technical member of staff, whereas a per-product contact may for example be a member of the publisher's marketing or publicity staff tasked with responsibility for the individual product. The contact details provided would clearly be *for trade use only*, and not intended for use within consumer-facing systems.

A new, optional and repeatable <ProductContact> composite has been added within P.19. This composite contains a mandatory <ProductContactRole> with associated Codelist 198, plus <ProductContactIdentifier>, <ProductContactName>, <ContactName> and <EmailAddress> elements identical to their counterparts in <Sender>, giving details of a contact at the publisher. The same <ProductContact> composite can also be used within P.25 where the contact is provided by a publisher representative in a market (*eg* an agent or local publisher).

The suggested initial codelist entries for <ProductContactRole> are limited to 'promotional contact' and a contact for accessibility queries, but others can be added as required.

Because <ProductContact> is added to P.25, the existing elements <PromotionContact>, <TelephoneNumber>, <FaxNumber> and <EmailAddress> (of the publisher's representative) in P.25 have been deprecated, and whenever possible, similar information should be carried in the new <ProductContact> composite instead.

This change has no effect on existing ONIX data.

8. Change to cardinality of many text data elements

Previously, data elements such as <BiographicalNote> carried a *language* attribute, but were not repeatable. Thus, biographical notes for particular authors could only be provided in a single language. This change to the ONIX 3.0 schema makes it possible to repeat <BiographicalNote> and many other textual data elements, to deliver the data in multiple languages. If there is only a single <BiographicalNote>, the *language* attribute is optional as it was previously. If there are multiple repeats of <BiographicalNote>, each repeat must carry a different *language* attribute, so that a data recipient may make a sensible choice among the available options.

Equivalent changes have been made to:
- <MessageNote>
- <DeletionText>
- <ProductFormFeatureDescription>
- <ProductFormDescription>
- <BiographicalNote>
- <ContributorDescription>
- <WebsiteDescription>
- <ProfessionalPosition>
- <ContributorStatement>
- <EditionStatement>
- <ReligiousTextFeatureDescription>
- <IllustrationsNote>

- <AncillaryContentDescription>
- <SubjectHeadingText>
- <AudienceDescription>
- <Text>
- <FeatureNote>
- <CitationNote>
- <PrizeJury>
- <PublishingStatusNote>
- <SalesRestrictionNote>
- <MarketPublishingStatusNote>
- <PriceTypeDescription>

Note that this is not the same selection of textual data elements as can use XHTML markup (see the list of 24 data elements in Section 2. above), though there is a high degree of overlap between the two lists.

The net result is that metadata for a single product can be provided in several languages 'in parallel', within a single Product record. Recipients of ONIX data may be able to make use of multiple languages, or they may pick the language that is most useful for their requirements. This will be of use in international book trading, where for example, a publisher sells a product in its domestic market and in export markets that use different languages (where it

becomes a 'foreign language book'), and in countries where there are multiple languages in common use (*eg* Switzerland, India).

This change has no effect on existing ONIX data.

Note that textual data within contributor names and product titles are not repeatable in this way. <AlternativeName> and <AlternativeTitle> provide methods for delivering transliterated names and translated titles.

9. Introduction of new *textscript* attribute

The changes in 8. above cover the requirement for provision of metadata in multiple languages in parallel, but do not provide a method where a contributor name such as Fyodor Dostoyevsky might be provided both in Cyrillic and Latin script (ie as both 'Фёдор Достоевский' and 'Fyodor Dostoyevsky').

A new *textscript* attribute has been added to enable transliterations of names to be carried:

```
<Contributor>
        <ContributorRole>A01</ContributorRole>
        <PersonName>다니엘 돔샤이트-베르크</PersonName>
        <AlternativeName>
                <NameType>05</NameType>
                <PersonName textscript="Latn">Daniel Domscheit-
                        Berg</PersonName>
        </AlternativeName>
</Contributor>
```

The primary contributor name should always use the script used on the product (in this case Hangul [Korean script]), and the transliteration is provided as an alternative name. Name type 05 is a new code in List 18 indicating 'transliterated form of primary name'.

Alternative names may themselves also occur in multiple scripts (*eg* two forms of a former name). In this case, any alternative name without a *script* attribute should use the same script as the primary name, and a transliteration of that alternative name should use the same <NameType> and carry a script attribute.

10. Other minor attribute changes

The <ThesisYear> data element is modified to take an optional *dateformat* attribute, so it may carry a full date if required (rather than just a year). Note that a thesis date should indicate the date of *acceptance* of a thesis, rather than its *submission*. <MessageNote> and <AudienceCodeTypeName> take an optional *language* attribute for consistency with other textual data elements.

11. Deprecation of the <Reissue> composite

The intent of the <Reissue> composite was to provide a way for suppliers to deliver information about a planned reissue of an existing product. However, *all information that might be carried in the composite can also be carried in other, more appropriate parts of the message*.

A 'reissue' occurs when a publisher makes an existing product available (again) with revised collateral material, a redesigned cover *etc* – but without significant changes to the content of the product and without any change in the product identifiers. This might occur regularly with perennial backlist products where the product receives a regular 'refresh' such as a

redesigned cover. The old and new versions are not available concurrently (since that would require them to be treated as two separate products, with separate identifiers), and there may or may not be an intervening period during which neither version of the product is available.

The challenge that faces metadata suppliers and recipients is to achieve an orderly switchover from using the old collateral material to using the new, revised material – and timing this switchover so that a consumer ordering the product on the basis of seeing, say, a cover image on a retailer's website, gets the product bound with the cover they expect.

The most appropriate ONIX 3 techniques for handling metadata associated with reissues are described in detail in the ONIX *Implementation and Best Practice Guide*, and they avoid use of the <Reissue> composite altogether. The simple date that a reissue is planned for should be carried in <PublishingDate> in P.20 and/or in <MarketDate> in P.25, and old and new collateral material in <TextContent>, <CitedContent> or <SupportingResource> should be associated with end and start dates (respectively) using <ContentDate>. The <Reissue> composite is deprecated to encourage use of the best practice.

This change would have no effect on existing ONIX data, but ONIX users are encouraged to use the methods outlined in the *Guide*.

12. Deprecation of <AudienceCode>

The <AudienceCode> element is redundant, as the same data can be provided within the <Audience> composite:
```
    <Audience>
        <AudienceCodeType>01</AudienceCodeType>
        <AudienceCodeValue>06</AudienceCodeValue>
    </Audience>
```
replaces:
```
    <AudienceCode>06</AudienceCode>
```
This change has no effect on existing ONIX data, but ONIX 3 users are encouraged to use the more generalized method with the <Audience> composite.

13. Change to datatype of <SubjectSchemeIdentifier>

List 27 has nearly exhausted the range of codes available. Ongoing growth in global use of ONIX, and development of a new version of the BIC subject scheme – which may require assignment of six or more new codes – means that it's likely that by the second half of 2012, no new codes will be available to identify new subject schemes.

Currently, codes in list 27 are defined as 'two digits', and are limited to the range 00–99. List 27 is redefined to contain to 'two *alphanumeric characters'*, extending the number of codes from 100 to more than 1000, and providing adequate 'headroom' for future new subject schemes. This would allow assignment of codes such as 0A, 0B… up to 0Z, 1A up to 9Z, and A0 to ZZ. (NB this is not an indication of the order in which codes will be assigned.)

Note that this change also applies to ONIX 2.1 and to Codelist 26 – at least for those recipients who need to create or accept any new codes. The first 'new-style' code in List 26 or 27 will not be assigned until at least July 2012.

Why extend to letters rather than increasing to three digits? Because where ONIX implementers have stored these codes at all, they have most likely used a string (a text-style data column) in their database rather than a number (*eg* `char(2)` rather than `integer`) – in part because of the leading zeros and in part because other codelists use letters as well as

digits. Anyone who has used a string datatype for storing the codes will not need to implement any change at all unless they need to handle a subject scheme that is assigned one of the new code numbers. If the length of the code were changed to three decimal digits, then many implementers would need to modify their database tables and their applications, even if they had no need to handle the new subject scheme codes. Use of two alphanumeric characters minimises the work required for everyone.

14. Change of namespace for ONIX for Books Release 3.0

Until late 2011, the documented XML 'namespace' for ONIX 3.0 was "http://www.editeur.org/onix/3.0/reference" (or "/short"). Messages may optionally include the namespace declaration near the start of the file:

```
<ONIXMessage release="3.0"
xmlns="http://www.editeur.org/onix/3.0/reference">
```

Such namespaces are increasingly important, particularly with the growth in use of RDF and Linked Data. Unfortunately, the use of 'www' as a part of the namespace creates an expectation that namespaces resolve to a web page – they often don't, and namespaces are intended purely as a method of creating a unique naming scheme for the data elements defined in the schema. As a result, EDItEUR is rationalising the use of namespaces in its various XML-based standards.

The new namespace is 'http://ns.editeur.org/onix/3.0/reference' (or '/short'), with the 'ns' serving to highlight the fact that this is a namespace only, and should not be expected to resolve. This change has already been implemented: ONIX 3.0 schemas included in downloadable ZIP packages with Codelists issue 15 already include versions with 'old' and 'new' namespaces. From Issue 16, only the new namespace will be used.

Note that when validating an ONIX message, it is common to add lines like this:

```
<ONIXMessage release="3.0"
xmlns="http://ns.editeur.org/onix/3.0/reference"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ns.editeur.org/onix/3.0/reference
ONIX_BookProduct_3.0_reference.xsd">
```

The namespace (in green) is specified with the *xmlns* attribute, and the last attribute *xsi:schemaLocation* links the namespace (in blue) to the actual location of the schema file (in red). The location in red needs to be adjusted to point to your own copy of the schema file on a local disc or perhaps on an intranet, since there is no copy hosted publicly on the Internet.

Some other ONIX-family messages already incorporate the new-style namespaces, including the latest revisions of ONIX for ISTC Registration and ONIX for ISBN Registration (the latter in fact uses the 'http://ns.editeur.org/onix/3.0/reference' namespace, as it is in effect a subset of ONIX for Books 3.0. **The namespaces for ONIX 2.1 will not be changed.**

The old namespaces should be treated as synonyms of the new namespaces, so there is no real effect on existing ONIX data. ONIX messages do not usually carry the *xmlns* or *xsi:schemaLocation* attributes, though the attributes might be added temporarily during internal processing and validation. Implementers may need to adjust their arrangements for validation to take account of the new namespace.

15. Naming and migration

These changes constitute Revision 1 of ONIX for Books 3.0 (also known as 3.0.1). All current ONIX messages remain valid under the revised schema, so there is no requirement to maintain original and updated schemas in parallel – all ONIX 3.0 users should update to the latest documentation, and data providers should update to the latest schema files and

codelists for validation purposes even if they do not intend to use any of the new features introduced in 3.0.1. Data recipients should update to the latest schema files as soon as practicable, even if they cannot immediately process any of the new data elements or attributes – they should simply ignore any elements they cannot make use of.

Graham Bell
EDItEUR
27 January 2012