EDITEUR

ONIX for Books

Product Information Message

Application Note: Block updates in ONIX 3.0

A key part of the design of ONIX 3.0 is the ability to use 'block updates' [1].

The normal way for a recipient to handle an ONIX message is that – in principle – each Product record can be processed individually, one record at a time, extracting each data element and updating the recipient's database. Each Product record starts with a unique and persistent <RecordReference> which should be used to check whether this Product record is an entirely new product, or an update of previously-sent information [2]. If it's new (*ie* the Record reference has never been encountered before), a new product is created into the recipient's database. If it's an update (*ie* the Record reference *has* been seen before), then the newly-supplied data completely replaces the previous information for that product – even when this implies deletion of some older data that is not present in the updated record.

But for block updates, the processing is different. The Product record is broken into seven 'blocks', and for each product, each *block* can be processed individually, replacing any previous information supplied *in that block* but leaving previous product information from other blocks intact.

A block update Product record in the ONIX message can consist of an informally-named 'block zero' – which contains data about the source of information in the record plus the product identifier(s) – plus any selection of the other six blocks. In effect, any of the six main blocks can be updated independently of the data in the other blocks [3]. And at least in principle, there is no particular problem in mixing full records with block updates in the same ONIX message – though data recipients might in some cases prefer to receive each type in separate ONIX messages.

Block updates were added to the ONIX 3.0 Specification in order to reduce the size of ONIX files, and reduce the amount of redundant data processing by recipients where most of the data in the supplied Product record is unchanged. They enable simpler price and availability updates, because all pricing and supplier availability information is in Block 6, which can be sent (along with block zero) without re-sending all the other bibliographic and marketing material. However, not all ONIX recipients can yet process block updates, so ensure they are supported by downstream data partners before sending.

The simplest possible block update

The simplest-possible block update would be an update with none of the six main blocks. Block zero is always required. Here's a more-or-less realistic example, where the update might be 'adding' a co-publisher's ISBN that was not included in the previously-sent ONIX:

```
<Product>
    <RecordReference>com.mypublisher.onix.01734</RecordReference>
    <!-- record source details omitted for clarity -->
    <NotificationType>04</NotificationType>             <!-- block update -->
    <ProductIdentifier>
        <ProductIDType>03</ProductIDType>                    <!-- GTIN-13 -->
        <IDValue>9780001234567</IDValue>
    </ProductIdentifier>
```

[1] block updates are not possible in earlier versions of ONIX

[2] using <RecordReference> to 'recognise' updates is strongly preferred to using a <ProductIdentifier> such as an ISBN. This allows the recipient to handle data about the same product but from different sources in a better-managed way, and allows errors with identifiers to be corrected without great effort

[3] note that as well as *updating* previously-supplied information, this implies the ability to *delete* almost any (or for some blocks, all) the information in a block

```
        <ProductIdentifier>
            <ProductIDType>15</ProductIDType>                    <!-- ISBN-13 -->
            <IDValue>9780001234567</IDValue>
        </ProductIdentifier>
        <ProductIdentifier>
            <ProductIDType>24</ProductIDType>       <!-- new co-publisher ISBN -->
            <IDValue>9789991234564</IDValue>
        </ProductIdentifier>
    </Product>
```

Note that the record does not contain any of <DescriptiveDetail>, <CollateralDetail>, <ContentDetail>, <PublishingDetail>, <RelatedMaterial> or <ProductSupply>. These are the six main blocks, and each is optional in a block update.

The effect of this block update should be that the bibliographic, marketing, and supplier and price information held by the ONIX recipient (which was received in previous ONIX messages) is retained, unchanged, and the additional co-publisher ISBN (which was not in the previous ONIX) is *added* to the recipient's database.

> In contrast, if the message above was *not* a block update (it would have different Notification type), then such a message consisting only of block zero implies *deletion* of all information previously held, irrespective of which block it was supplied in, and replacement with *only* the data supplied in the new message. All that would remain in the recipient's database would be the supplied GTIN and ISBNs. (This ignores the fact that this hypothetical ONIX would be invalid, because Blocks 1 and 4 are mandatory, and Block 6 is expected to be included too).

Single-block block updates

In the period leading up to publication, it's common for there to be frequent updates of the metadata – for example providing extra marketing material, adding the biography or professional affiliation of an author, or updating the expected publication date. Each time – without using block updates – all the existing information needs to be included in the Product record, whether it's unchanged or new. This means larger files need to be exchanged, and it requires a great deal of unnecessary processing by recipients. *With* block updates, only those blocks that contain new of modified metadata need be sent and processed. An extra bit of marketing material – send Block 2. Change to some contributor information – Block 1. An updated publication date – Block 4.

A block update that merely updates the publication date – where all the other metadata about the book is unchanged – could look like this, including the mandatory block zero and Block 4:

```
    <Product>
        <RecordReference>com.mypublisher.onix.01734</RecordReference>
        <NotificationType>04</NotificationType>              <!-- block update -->
        <ProductIdentifier>
            <ProductIDType>03</ProductIDType>                    <!-- GTIN-13 -->
            <IDValue>9780001234567</IDValue>
        </ProductIdentifier>
        <ProductIdentifier>
            <ProductIDType>15</ProductIDType>                    <!-- ISBN-13 -->
            <IDValue>9780001234567</IDValue>
        </ProductIdentifier>
        <PublishingDetail>                                       <!-- block 4 -->
            <Imprint>
                <ImprintName>MyBooks</ImprintName>
            </Imprint>
            <Publisher>
                <PublishingRole>01</PublishingRole>
                <PublisherName>My Publisher</PublisherName>
```

```
            </Publisher>
            <CityOfPublication>Springfield, IL</CityOfPublication>
            <CountryOfPublication>US</CountryOfPublication>
            <PublishingStatus>02</PublishingStatus>          <!-- forthcoming -->
            <PublishingDate>
                <PublishingDateRole>01</PublishingDateRole>      <!-- pub date -->
                <Date>20190926</Date>                        <!-- now Sept 26 -->
            </PublishingDate>
            <SalesRights>
                <SalesRightsType>01</SalesRightsType>            <!-- for sale -->
                <Territory>
                    <RegionsIncluded>WORLD</RegionsIncluded>   <!-- everywhere -->
                </Territory>
            </SalesRights>
        </PublishingDetail>
    </Product>
```

Block 4 is highlighted in red, and the only updated element is picked out in green. It's not possible to send *just* the new publication date – the whole of the block must be included. Even with a block update, there is still an 'overhead' of unchanged metadata that needs to be sent and processed, but it is *much less* than the complete record that would otherwise have to be sent.

What happens if a sender does not include all the unchanged metadata in the block? This implies that previously held data should be deleted by the recipient. And this is sometimes vital with updates that involve repeated composites. Say that you expect to publish a book with joint authorship – three contributors working together. But one drops out and her contributions will not be included. The original ONIX listed three contributors, and so you could provide a block update containing Block 1 (plus block zero). The updated Block 1 would list only two contributors. For the recipient, this means that all previously-held Block 1 metadata should be deleted and replaced with the new updated Block 1 metadata – in effect, three contributors would be replaced by two.

Price and availability updates

In the period after publication, it's unlikely that information in Block 1 would need updating – unless for example a list of keywords (in <Subject>) is revised. Block 2 may need updates as reviews are gathered. Block 4 might only need updating as the product reaches the end of its life and the Publishing status is revised to 'out of print'. But Block 6 – which contains the distribution and supply details such as availability and pricing – is likely to require frequent updates throughout the product's life. A block update provides a relatively efficient way of distributing changes to the product's price and availability [4] [5].

Below is an example of a price and availability update as a Block 6-only block update. In this case, the update is limited to the US Price status and Price amount – which might previously have been a provisional $10.95 and is updated in this message to a firm price of $9.95:

---

[4] ONIX 2.1 specified a special abbreviated variation of the message for the same purpose (the *ONIX 2.1 Supply Update* message). But an ONIX 3.0 block update is *not* a special variation – it is a 'normal' ONIX Product record and there is no requirement for a specialised price and availability update message format

[5] Price and availability updates are often delivered using quite different *non*-ONIX messages, for example using the EDItX, EDIFACT, X.12 or TRADACOMS standards. In such cases, it's important to make the switch from one standard to another at the right point in the product lifecycle. And since ONIX updates are unlikely to stop completely (because of the need to update metadata *other than* price and availability), it's vital to avoid inconsistent data being provided to the supply chain in the two separate communication channels. Note that some details of complex pricing are possible in ONIX and EDItX but cannot be expressed in EDIFACT, X.12 or TRADACOMS

```
<Product>
    <RecordReference>com.mypublisher.onix.01734</RecordReference>
    <NotificationType>04</NotificationType>              <!-- block update -->
    <ProductIdentifier>
        <ProductIDType>03</ProductIDType>                    <!-- GTIN-13 -->
        <IDValue>9780001234567</IDValue>
    </ProductIdentifier>
    <ProductIdentifier>
        <ProductIDType>15</ProductIDType>                    <!-- ISBN-13 -->
        <IDValue>9780001234567</IDValue>
    </ProductIdentifier>
    <ProductSupply>                                          <!-- block 6 -->
        <SupplyDetail>
            <Supplier>
                <SupplierRole>01</SupplierRole>
                <SupplierName>My Distributor</SupplierName>
            </Supplier>
            <ProductAvailability>10</ProductAvailability>     <!-- not yet -->
            <SupplyDate>                                    <!-- available -->
                <SupplyDateRole>08</SupplyDateRole>
                <Date>20190923</Date>              <!-- expected ship date -->
            </SupplyDate>
            <Price>                              <!-- dollar price for USA -->
                <PriceType>01</PriceType>                <!-- price ex tax -->
                <PriceStatus>02</PriceStatus>         <!-- new firm status -->
                <PriceAmount>9.95</PriceAmount>       <!-- new lower price -->
                <CurrencyCode>USD</CurrencyCode>
                <Territory>
                    <CountriesIncluded>US</CountriesIncluded>
                </Territory>
            </Price>
            <!-- other price composites omitted -->
        </SupplyDetail>
    </ProductSupply>
</Product>
```

Note that some data elements that are normally included within the Product supply composite – such as any market-specific details in <Market> and <MarketPublishingDetail>, a supplier identifier, returns information, discounts and so on – have been omitted from the example for the sake of simplicity. In a real price and availability block update, the whole of Block 6 must be supplied – including full market detail and multiple prices (if there are multiple prices across different territories within the market).[6]

Multi-block block updates

Of course it's possible to update more than one block at a time – a block update could contain Blocks 1 and 4 if there are changes in both blocks. And if *all* relevant blocks include changes, it's no longer a block update… and Notification type would then no longer be 04.

[6] special care is needed if there are multiple markets (that is, repeats of <ProductSupply>. If the original message contained two <ProductSupply> composites, then the block update should contain two <ProductSupply> composites too – even if only one of them contains any updated metadata. Failing this, information relating to the 'other' market (ie the unchanged market) may well be deleted by the recipient

Deletion messages

Another real-world example that can *look like* a simple Block update is a 'deletion request', sent for example when a Product record has been distributed in error. Note the different Notification type:

```
<Product>
    <RecordReference>com.mypublisher.onix.01734</RecordReference>
    <NotificationType>05</NotificationType>                <!-- deletion -->
    <DeletionText>Record issued in error</DeletionText>
    <ProductIdentifier>
        <ProductIDType>03</ProductIDType>                   <!-- GTIN-13 -->
        <IDValue>9780001234567</IDValue>
    </ProductIdentifier>
    <ProductIdentifier>
        <ProductIDType>15</ProductIDType>                   <!-- ISBN-13 -->
        <IDValue>9780001234567</IDValue>
    </ProductIdentifier>
</Product>
```

This is *not* a block update, because it implies deletion of *all* metadata for the supplied Record reference, rather than simple updating of the block zero information (leaving the other data unchanged) as would be the case with a Notification type 04 block update. Notification type 05 requests complete deletion the product in the recipient's database [7].

Note however that such a deletion should NOT be used in cases where a product has been abandoned prior to publication, or where the product has become unavailable or out of print – these cases are a simple change of Publishing status or Product availability, and data recipients need to retain records to deal with ongoing customer enquiries. Because data senders have historically *mis*-used deletion requests for abandoned or out of print titles, many data recipients are often forced to ignore legitimate deletion requests.

Graham Bell
EDItEUR
3rd October 2019

[7] it's vitally important to use the Record reference for matching a deletion request to previously-supplied data as described on page 2. One potential reason for deletion is the inadvertent use of a duplicate ISBN or error in assignment of a Product identifier, and use of the identifier for matching could easily lead to deletion of the wrong product