EDITEUR

ONIX for Books

Product Information Message

Application Note: Pricing in ONIX 3.0, part 2

The section of ONIX that defines the price of a particular product is – for many users of the standard – the most obvious single reason for adopting ONIX. The standard is designed to support unambiguous communication of *product* metadata through the supply chain. While ONIX is used widely for exchange of purely bibliographic data, it is expressly designed for *commercial* use cases.

It is an important point: an ONIX record is both a description of the product *and* a commercial offer, which includes pricing and various terms of trade – returns arrangements, restrictions on sale and so on.

This document should be read in conjunction with *Pricing in ONIX 3.0, part 1*.

Advanced pricing

As noted in part 1, a commercial model used widely across the industry (though not in all countries) is based on a stated retail price and a trade discount applied to business-to-business transactions. So while part 1 showed how retail prices should be specified in ONIX, the full information required by the supply chain includes the trade discount on offer to the purchaser (typically a wholesaler or retailer).

In the most simple cases, a supplier may set a single discount – 32.5% or whatever – and this would indicate that all trade purchasers could buy a copy for 32.5% less than the applicable retail price. In more complex cases, the supplier may wish to specify a higher discount for some customers and a lower discount for others.

Discounts

If the supplier wants to set a single, simple discount percentage, this can be as straightforward as:

```
<Discount>
    <DiscountPercent>32.5</DiscountPercent>
</Discount>
```

But this simple case may not be enough. It is common to set a discount that rises according to the order quantity – purchase more, and the buyer gets better commercial terms. The <Quantity> data element specified the number of copies needed to qualify for a particular level of discount:

```
<Discount>
    <DiscountPercent>32.5</DiscountPercent>
</Discount>
<Discount>
    <DiscountType>01</DiscountType>                    <!-- rising discount -->
    <Quantity>10</Quantity>
    <DiscountPercent>36.5</DiscountPercent>
</Discount>
<Discount>
    <DiscountType>01</DiscountType>
    <Quantity>20</Quantity>
    <DiscountPercent>38.5</DiscountPercent>
</Discount>
```

In this case, the purchaser ordering fewer than 10 copies gets a discount of 32.5% on their order, whereas ordering 10–19 copies qualfies for an additional 4% discount and ordering 20 or more copies qualfies for a further 2%. Note that the discount is applied across all the copies ordered, so an order for 25 copies would cost 25 x retail price (excluding any retail-level taxes) x 0.615 (*ie* 38.5% discount).[1]

[1] Unfortunately, rising discounts can encourage deliberate over-ordering and subsequent returns, in order to qualify for a higher discount on the copies that are not returned, unless returns processing reduces credits to account for the fact that the discount on the original order would have been lower if the correct number of copies had been ordered. This does not apply so strongly to progressive discounts

Another option is that the higher discount applies only to the copies at or above the qualifying minimum [2]:

```
<Discount>
    <DiscountPercent>32.5</DiscountPercent>
</Discount>
<Discount>
    <DiscountType>03</DiscountType>                    <!-- progressive discount -->
    <Quantity>10</Quantity>
    <DiscountPercent>36.5</DiscountPercent>
</Discount>
<Discount>
    <DiscountType>03</DiscountType>
    <Quantity>20</Quantity>
    <DiscountPercent>38.5</DiscountPercent>
</Discount>
```

Here, a similar order for 25 copies would cost slightly more (9 x price x 0.675) + (10 x price x 0.635) + (6 x price x 0.615). The lowest discount applies to the first nine copies, then copies 10–19 qualify for the middle level of discount and copies 20–25 qualify for the highest discount. The effective discount across the 25 copies is 35.54%.

Discount codes

Advertising a discount as above implies the percentages are the same for all customers. This may not be the case. And when discounts vary from customer to customere, they may well be commercially very sensitive. So there is a way to deliver unique discounts to individual customers without revealing the percentages to all ONIX recipients – while still sending exactly the same data to all. This relies on *discount group codes*:

> A supplier (publisher, distributor or wholesaler *etc*) establishes several internal 'discount groups', numbered or lettered. Each product is allocated to a group. The groups do not relate directly to actual discount percentages, but contain many similar books for which the discount will be the same. The supplier then agrees a private lookup table of actual discounts with each retailer customer – in the table, each group is assigned a discount percentage, but the table itself can be unique to each customer. This way, the groups can remain the same while the actual discount for each group can be different for each customer, and can vary over time.
>
> So distributor A might have three groups, 01, 02 and 03. For A's retailer customer X, each group maps to a discount – maybe group 01 means 45%, group 02 means 49% and group 03 means 52%. For A's customer Y, each group might map to a *different* discount – group 01 = 47%, group 02 = 51% and group 03 = 55%.
>
> The publisher can include the *discount group* in the metadata for each product, without revealing the actual percentage discounts to everyone. And for any one product or group of products, different discounts may apply to each customer with the minimum of fuss – each customer has a unique lookup table that lists the discount percentage for each group.

But inevitably, *every* supplier has groups 01, 02 and 03, so the discount groups must be identified using a unique code. This can be proprietary or standardised.

```
<DiscountCoded>
    <DiscountCodeType>02</DiscountCodeType>              <!-- proprietary scheme -->
    <DiscountCodeTypeName>Acorn Press DG</DiscountCodeTypeName>
    <DiscountCode>03</DiscountCode>
</DiscountCoded>
```

In this case, it's important the supplier keeps the <DiscountCodeTypeName> consistent, because the ONIX recipient may have many tables, from many suppliers, and *most* will contain a group 03. Effectively, the code type name tells the ONIX recipient which table to use.

In the UK, BIC (https://www.bic.org.ok) maintains a naming scheme for discount code groups. In effect, each supplier using the scheme is allocated a five letter prefix, then each publisher can add a

[2] Optionally, ONIX allows a <ToQuantity> to be specified as well, making the discount 'bands' more obvious, but this isn't necessary in a case as clear as this.

one to three character suffix (typically digits) for each group. Using this scheme, every discount group for every supplier using the scheme is guaranteed a unique code.

```
<DiscountCoded>
    <DiscountCodeType>01</DiscountCodeType>                    <!-- BIC scheme -->
    <DiscountCode>AWXYZ12</DiscountCode>
</DiscountCoded>
```

So the unique discount percentage applicable to an ONIX recipient can be looked up in the row for group 12 in the table for supplier 'AWXYZ'.

Of course, there is no particular reason why a group implies a simple percentage (albeit a different percentage for each customer). A group could relate to an agreed rising or progressive discount based on order quantities too. And for agency pricing, commission groups can be handled in the same way (though it is much less common to maintain multiple commission levels).

Similar schemes may operate in other countries.

Do note that these discounts can be described as the 'base terms' for transactions. A supplier may also apply retrospective discounts based on volumes ordered over a period.

Returns

In the book trade, the 'sale or return' model is common: unsold books may be returned to the supplier for full credit. But there are some nuances. A significant group of books – particularly those that are manufactured to order – are 'firm sale'. That is, once purchased, they cannot be returned for credit. Others can be 'returned', but only the cover need be stripped off and physically sent back to prove the copy has been rendered unsaleable.

The returnability of stock is an important part of the commercial offering described in an ONIX record, though it is not set within <Price>, as it tends to be set by the supplier on a per-product basis [3]:

```
<ReturnsConditions>
    <ReturnsCodeType>04</ReturnsCodeType>                  <!-- ONIX list 204 -->
    <ReturnsCode>02</ReturnsCode>                              <!-- firm sale -->
</ReturnsConditions>
```

ONIX List 204 includes the most common returns conditions, including goods on consignment. The next issue of the codelists will (most likely) allow differentiaton between goods that are returnable (and which can subsequently be re-sold by the supplier) and copies which can be stripped or destroyed by the customer. A <ReturnsNote> may also be included, for example to give details of any returns authorisation process that must be followed – for example 'call Customer Service for authorisation before returning'.

Free of charge products

It is a fairly common error to list free of charge products with a price of 0.00 – yet it is also clear that this has never been the correct way to specify a product is free. ONIX has always used <UnpricedItemType> as an alternative to <Price> or <PriceAmount>. Unfortunately, DTD validation cannot highlight a zero price amount as invalid, and this may have led to the erroneous belief that zero prices are okay.

The only correct way to specify a product is free of charge is to use <UnpricedItemType>.

Having said that, in ONIX 3.0, the Unpriced item type data element can be used in either of two ways – and for each product, it's pretty simple to choose between them. The first option – using <UnpricedItemType> instead of a <Price> – is used when the product is free of charge *in all circumstances*. This could be the case for open access products or promotional items. The second – <UnpricedItemType> instead of <PriceAmount> – allows all the other features of <Price> to be used at

[3] This means you cannot easily set one price for sale-or-return terms and a different price for firm-sale terms. To do so would require a repeat of <SupplyDetail> or <ProductSupply>, rather than a repeat of <Price>. This may be necessary when goods sold to the domestic market are returnable and goods sold internationally are not, but within a domestic market, it is rare: sales of indistinguishable copies of a product on both firm-sale and sale-or-return terms invite abuse of the returns system.

the same time, so a product can be free in one country but priced conventionally in another, or can be free from or until a certain date.

```
<SupplyDetail>
    <Supplier>
        . . .                                    <!-- details omitted for brevity -->
    </Supplier>
    <ProductAvailability>21</ProductAvailability>                <!-- in stock -->
    <UnpricedItemType>01</UnpricedItemType>          <!-- always free of charge -->
</SupplyDetail>
```

Or…

```
<Price>
    <PriceType>02</PriceType>
    <UnpricedItemType>01</UnpricedItemType>                <!-- free of charge -->
    <Territory>
        <CountriesIncluded>IE</CountriesIncluded>              <!-- in Ireland -->
    </Territory>
    <PriceDate>
        <PriceDateRole>14</PriceDateRole>.               <!-- from 30th August -->
        <Date>20190830</Date>
    </PriceDate>
</Price>
```

The second option would normally be accompanied by more conventional prices that are relevant ouside Ireland, or inside Ireland but prior to the end of August.

EDItEUR
22/07/2019

These notes and the accompanying part 1 are adapted from the *ONIX 3.0 Implementation and Best Practice Guide* (DOI: 10.4400/zuim).