DIGITALISERINGSSTYRELSEN

OIOUBL Guideline

OIOUBL Prices

UBL 2.0 Priser

G25

Version 1.3

Colophon

Contact:

Danish Agency for Digitisation

E-mail: support@nemhandel.dk

OIOUBL Version 2.02
July 2015
Danish Agency for Digitisation

Landgreven 4
DK-1017 Copenhagen
Phone +45 3392 5200
http://www.digst.dk
digst@digst.dk

Contents

1. Preface

These guidelines form of a series describing the purpose and use of the business documents that comprise the Danish localization of UBL 2.0, known as OIOUBL.

As well as guidelines describing the use of commonly used elements, a separate guideline has been prepared for each business document.

1.1. Purpose of this document

This guideline describes the use of classes and elements that deal with prices and quantities.

In this document special focus is given to:

- Elements (and how they interrelate) that are central for specifying prices and quantities
- How these elements may be used to define different price/quantity relationships

This guideline covers all documents that involve prices and quantities, but it is primarily relevant for ordering and invoicing related documents.

For further information about catalogue documents, refer to the specific OIOUBL guideline Catalogue Prices and Quantities (Ref. G40).

1.2. General Points

It should be possible to specify prices and quantities of items in documents in such at way that they match between orders and invoices. This means that it must be possible to transfer the definitions of prices and quantities from catalogue documents directly to the order documents, and subsequently to the invoice/credit note (and other documents).

Note that when prices (*PriceAmount*) are specified they are always exclusive of VAT.

1.3. Changes in version 1.3

In this latest update of this guideline the following has been changed:

- Questions and answers from FAQ on OIOUBL.info has been incorporated

2. Relevant UBL Classes and Elements

The fields which are relevant for specifying prices and quantities are placed directly under the line level of the respective documents, for example:

- *OrderLine*
- *InvoiceLine*

On the *OrderLine* this is primarily relevant for the following classes within the respective *LineItem*.

- *Quantity*
- *LineExtensionAmount*
- *Delivery (Quantity)*
- *Price (PriceAmount, BaseQuantity, OrderableUnitFactorRate)*
- *Item (PackQuantity, PackSizeNumeric)*

On *InvoiceLine* it primarily includes the classes

- *InvoicedQuantity*
- *LineExtensionAmount*
- *Delivery (Quantity)*
- *Item (PackQuantity, PackSizeNumeric)*
- *Price (PriceAmount, BaseQuantity, OrderableUnitFactorRate)*

2.1. DK element names and cardinality

The table below lists the elements and their names in Danish, as well as the cardinality.

2.1.1. OrderLine/LineItem

UK-name	DK-name	Use
Quantity	Mængde	1
LineExtensionAmount	VareLinjeBeløb	0..1
Delivery / Quantity	Levering / Mængde	0..1
Item / PackQuantity	Vare / PakkeMængde	0..1
Item / PackSizeNumeric	Vare / PakkeStørrelse	0..1
Price / PriceAmount	Pris / PrisBeløb	1
Price / BaseQuantity	Pris / BeregningsGrundlagMængde	0..1
Price / OrderableUnitFactorRate	Pris / OrdreAntalMængdeRate	0..1

2.1.2. InvoiceLine

UK-name	DK-name	Use
InvoicedQuantity	FaktureretMængde	1
LineExtensionAmount	LinjeTotal	1
Delivery / Quantity	Levering / Mængde	0..1
Price / PriceAmount	Pris / PrisBeløb	1
Price / BaseQuantity	Pris / BeregningsGrundlagMængde	0..1
Price / OrderableUnitFactorRate	Pris / OrdreAntalMængdeRate	0..1

2.1.3. Explanations of the most important elements

The following elements at line level are relevant for prices and quantities:

UK-name	DK-name	Use	Remarks
Quantity	Mængde	1	Number of units of the quantity in question.
Quantity@unitCode			Unit code for the Quantity. The value must be a valid unit of measure code. For example, ”CS” for case.
PackQuantity	PakkeMængde	0..1	The packing quantity of the Item. Contains the number of units as defined in PackSizeNumeric. For example, if the packing is a case of 12 pieces., PackQuantity should be ”1”, and Case should be specified as the unitCode attribute, as described below.
PackQuantity@unitCode			Specifies the unit for PackQuantity. The value must be a valid unit of measure code. For example, ”CS” for case.
PackSizeNumeric	PakkeStørrelse	0..1	The number of units in one pack of the Item in question. Using the previous example, this would be defines as ”12”.
PriceAmount	PrisBeløb	1	The net price for the BaseQuantity of the item. Use a full stop/period as decimal seperator.
PriceAmount@currencyID			The currency that applies to the price, expressed using a code, such as ”DKK”, ”EUR”, etc.
BaseQuantity	BeregningsGrundlagMængde	0..1	The base quantity applicable to the price. For example, if the price specifed in PriceAmount is DKK 65.00 for one bottle of wine, then BaseQuantity should be "1". The unit is specified in the unitCode attribute, as described below.*
BaseQuantity@unitCode			Specifies the unit code for the BaseQuantity. The value must be a valid unit of measure code. For example, ”BO” for bottle
OrderableUnitFactorRate	OrdreAntalMængdeRate	0..1	The calculation factor from the Base Unit in BaseQuantity to the orderable unit in the Quantity element.*

* Note that *BaseQuantity* and *OrderableUnitFactorRate* should be filled out, and that they will be given a default value if no value is assigned. *BaseQuantity* is given the default value ”1 EA” (each) and *OrderableUnitFactorRate* is given the default value ”1”.

In the *Price* class you will also find the field *PriceTypeCode.* By mistake this field is set to “Bilateral agreed” in the documents OIOUBL Catalogue, Order, OrderChange and OrderResponse.

It should be set to “Used”, meaning that the receiver should read the value.

In a Catalogue the element can be used to specify e.g. a list price (See guide OIOUBL_GUIDE_CATALOGUE_PRICE (G40) section 4.5). The element is also used to specify whether a price is inclusive or exclusive taxes (Not VAT). The element must always use the following code lists: <cbc:PriceTypeCode listAgencyID=”6” listID=”UN/ECE 5387”/>.

Please notice that the element is only used if the OIOUBL default rules for pricing are deviated from. In OIOUBL the following code list values are relevant: “DR” (list price) and “ABE” (if the unit price is exclusive the specified non-VAT taxes). See the guide G27 about TAX.

3. Description

In the following section specific classes and elements related to prices and quantities will be described further.

3.1. Relationships between Price and Quantity

The figure below describes the overall relationship between Price and Quantity classes and their elements.

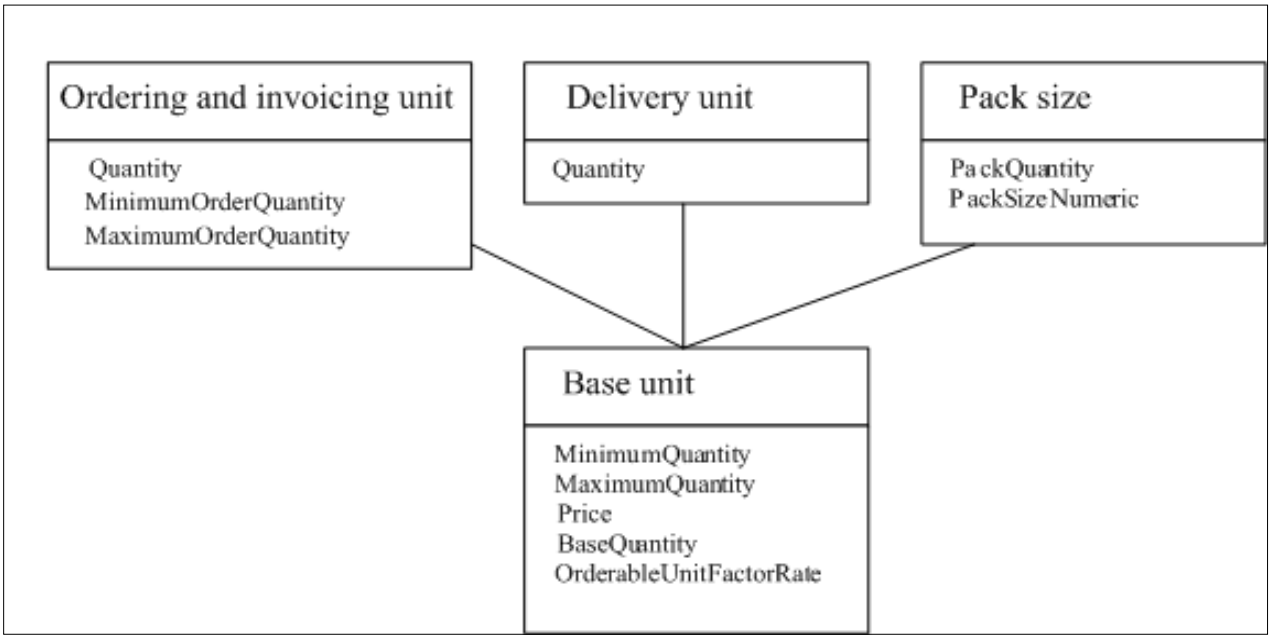

Figure 1: The relation between the Price and Quantity classes

The point of distinction is the difference in base units (including the supplier's base units (*BaseQuantity*) and base price (*Price*)). All orderable units and chargeable units are defined based on their respective base units. These elements and related information are specified in the *OrderLine* and *InvoiceLine* classes together with the information on delivery units from the Delivery class.

Note that it is also possible to define pack sizes for a given product in the product description.

3.2. BaseQuantity

When calculating prices and quantities, the *Price* class is the foundation for all other classes and elements.

The example below shows how the *Price* class may be specified:

```
<cac:InvoiceLine>
    <cbc:InvoicedQuantity unitCode="BO">12</cbc:InvoicedQuantity>
    <cbc:LineExtensionTotalAmount currencyID="DKK">720.00</cbc:LineExtensionTotalAmount>
    <cac:Item>
        <cbc:Name>Red wine</cbc:Name>
        <cac:SellersItemIdentification>
              <cbc:ID>1234567</cbc:ID>
        </cac:SellersItemIdentification>
    </cac:Item>
    <cac:Price>
        <cbc:PriceAmount currencyID="DKK">60.00</cbc:PriceAmount>
        <cbc:BaseQuantity unitCode="BO">1</cbc:BaseQuantity>
        <cbc:OrderableUnitFactorRate>1</cbc:OrderableUnitFactorRate>
    </cac:Price>
</cac:InvoiceLine>
```

Figure 2. Simple price example

The example shows that 12 bottles of wine have been invoice at a total price of DKK 720.00.

It also shows that each bottle has a price of DKK 60.00, and each orderable unit (*OrderableUnitFactorRate*) contains exactly 1 bottle.

The *PriceAmount* and the *BaseQuantity* express the supplier's base unit. That is, the units that the supplier maintains his goods in.

The more advanced example in Figure 3. shows that a case of Red wine has been invoiced at a price of DKK 720.00.

```
<cac:InvoiceLine>
     <cbc:InvoicedQuantity unitCode="CS">1</cbc:InvoicedQuantity>
     <cbc:LineExtensionTotalAmount currencyID="DKK">720.00</cbc:LineExtensionTotalAmount>
     <cac:Item>
          <cbc:Name>Red wine</cbc:Name>
          <cac:SellersItemIdentification>
               <cbc:ID>1234567</cbc:ID>
          </cac:SellersItemIdentification>
      </cac:Item>
      <cac:Price>
          <cbc:PriceAmount currencyID="DKK">60.00</cbc:PriceAmount>
          <cbc:BaseQuantity unitCode="BO">1</cbc:BaseQuantity>
         <cbc:OrderableUnitFactorRate>12</cbc:OrderableUnitFactorRate>
     </cac:Price>
</cac:InvoiceLine>
```

Figure 3. Advanced price example

This also shows that each bottle has a price of DKK 60.00, but now each orderable unit (*OrderableUnitFactorRate*) is 12 bottles.

This means that while this may be the same item shown in Figure 2, now the *OrderableUnit* differs from the *BaseQuantity*.

3.3. Delivery Unit

The example below shows how the units of delivery information is specified:

```
<cac:InvoiceLine>
  …
        <cac:Delivery>
                <cbc:Quantity unitCode="CS">1</cbc:Quantity>
        </cac:Delivery>
  …
</cac:InvoiceLine>
```

Figure 4. Example of a delivery unit

This shows that the *Quantity* the item is delivered in is cases (CS).

In the *Delivery* class it is not possible to identify a relation between the *BaseQuantity* and the delivered quantity. The details about delivery units (*Delivery*) are defined in the *InvoiceLine* class, and must follow this relationship.

3.4. Orderable and Invoice unit

In the following the order and invoice units are described.

3.4.1. OrderableUnit

Information about ordering items is specified within an *OrderLine* using the *LineItem* class, as shown in the example below:

```
<cac:OrderLine>
…
    <cac:LineItem>
          <cbc:ID>1</cbc:ID>
          <cbc:Quantity unitCode="CS">1</cbc:Quantity>
          <cbc:LineExtensionAmount currencyID="DKK">720</cbc:LineExtensionAmount>
          <cac:Price>
                <cbc:PriceAmount currencyID="DKK">60.00</cbc:PriceAmount>
                <cbc:BaseQuantity unitCode="BO">1</cbc:BaseQuantity>
                <cbc:OrderableUnitFactorRate>12</cbc:OrderableUnitFactorRate>
          </cac:Price>
          <cac:Item>
                <cbc:Name>Red wine</cbc:Name>
                <cac:SellersItemIdentification>
                      <cbc:ID>1234567</cbc:ID>
                </cac:SellersItemIdentification>
          </cac:Item>
```

```
   </cac:LineItem>
        …
<cac:OrderLine>
```

Figure 5. Example of OrderableUnit

The *ID* uniquely identifies the relevant line item on the Order.

There is a direct relationship between *LineExtensionAmount, Quantity, PriceAmount, BaseQuantity,* and *OrderableUnitFactorRate*.

This can be expressed as:

*BaseQuantity * OrderableUnitFactorRate = the quantity specified by Quantity@unitCode*

For example, if the *BaseQuantity* is "1 BO (bottle)", *OrderableUnitFactorRate* is "12", and the *Quantity@unitCode* is "CS", then the order quantity is "1 case of 12 bottles".

The price for the orderable unit is calculated likewise, such that:

*PriceAmount / BaseQuantity * (BaseQuantity * OrderableUnitFactorRate) = the price of one orderable unit.*

This expression can be reduced to:

*PriceAmount * OrderableUnitFactorRate = the price of one orderable unit.*

For example, if *PriceAmount* is DKK 60.00, *BaseQuantity* is "1" and *OrderableUnitFactorRate* "12", then the *LineExtensionAmount* is DKK 720.00 for a 12 bottle case (which is the *Orderable Unit*).

3.4.2. InvoicedQuantity

A similar logic to that of orderable units applies to invoiced quantities.

```
<cac:InvoiceLine>
  …
     <cbc:ID>1</cbc:ID>
     <cbc:InvoicedQuantity unitCode="BO">12</cbc:InvoicedQuantity>
     <cbc:LineExtensionAmount currencyID="DKK">720</cbc:LineExtensionAmount>
     <cac:Item>
          <cbc:Name>Red wine</cbc:Name>
          <cac:SellersItemIdentification>
               <cbc:ID>1234567</cbc:ID>
          </cac:SellersItemIdentification>
     </cac:Item>
     <cac:Price>
          <cbc:PriceAmount currencyID="DKK">60.00</cbc:PriceAmount>
          <cbc:BaseQuantity unitCode="BO">1</cbc:BaseQuantity>
          <cbc:OrderableUnitFactorRate>1</cbc:OrderableUnitFactorRate>
     </cac:Price>
       …
<cac:InvoiceLine>
```

Figure 6. Example of InvoicedQuantity

The *ID* uniquely identifies the relevant invoice line on the invoice.

In this example the *BaseQuantity* and the *InvoicedQuantity* are identical. This means the *LineExtensionAmount* is calculated as:

*PriceAmount / BaseQuantity * (BaseQuantity * OrderableUnitFactorRate) * InvoicedQuantity*

or

DKK 60.00 / 1 (1* 1) * 12 = DKK 720.00*

3.5. PackSizeNumeric

Two other elements are related to the specification of units. These are the *PackQuantity* and the *PackSizeNumeric*. Both are specified in the definition of an item.

```
<cac:Item>
        …
        <cbc:PackQuantity unitCode="CS">1</cbc:PackQuantity>
        <cbc:PackSizeNumeric>12</cbc:PackSizeNumeric>
        …
<cac:Item>
```

Figure 7: Example of PackQuantity and PackSizeNumeric

The specification for packs is only found within the Item class, but it must be considered in respect to the other unit specifications. That is, *PackQuantity* may be an expression of the packing (in the example, "1 case (CS)"). And *PackSizeNumeric* specifies how many items comprise the package (in the example "12").

PackSizeNumeric is related to *BaseQuantity*, as in the expression:

*BaseQuantity * PackSizeNumeric = the quantity expressed by PackQuantity@unitCode.*

Where *PackQuantity* is the quantity that is contained in a pack.

3.6. Decimals and roundings

Notice that there are no limits on the number of decimals on the unit price (*Price/PriceAmount*), but on the line extension amount (*InvoiceLine/LineExtensionAmount*) only 4 decimals are allowed.

To avoid large differences in the amounts it is recommended, that the sender of the document uses as many decimals as possible on *PriceAmount, BaseQuantity* and *OrderableUnitFactorRate*. To few decimals can cause large differences as in the example below, e.g:

Quantity	Price	Line total
10000	1.02	10200.00
10000	1.204	10240.00

See the guide on Totals (G28) for more information on decimals and rounding.

4. Examples

This section contains different examples of how to use the price and quantity elements, as well as the relationship between them.

4.1. Converting BaseQuantity to InvoicedQuantity

In some products and industries the *OrderableUnit* and the *InvoicedQuantity* for an item may not be the same. For example, oil is ordered in barrels but charged per liter, meat is ordered by cut but charged by weight, and steel is ordered by length measure but also charged by weight.

In the OIOUBL documents it is possible to manage these situations.

The following example show par of an invoice for one barrel of oil (containing 750 litres). These is the orderable units. However, the supplier's base quantity (for charging) is liters.

```
<cac:InvoiceLine>
  …
     <cbc:ID>1</cbc:ID>
     <cbc:InvoicedQuantity unitCode="BLL">1</cbc:InvoicedQuantity>
     <cbc:LineExtensionAmount currencyID="DKK">3600.00</cbc:LineExtensionAmount>
     <cac:Item>
          <cbc:Name>Let smøreolie</cbc:Name>
          <cac:SellersItemIdentification>
               <cbc:ID>11223344</cbc:ID>
          </cac:SellersItemIdentification>
      </cac:Item>
      <cac:Price>
           <cbc:PriceAmount currencyID="DKK">4800.00</cbc:PriceAmount>
           <cbc:BaseQuantity unitCode="LTR">1000</cbc:BaseQuantity>
           <cbc:OrderableUnitFactorRate>0.75</cbc:OrderableUnitFactorRate>
     </cac:Price>
       …
<cac:InvoiceLine>
```

Figure 8: Example of conversion of units

The quantity unit code of "BLL" in *InvoicedQuantity* specifies that the invoiced quantity is a barrel.

The supplier's base price *(PriceAmount*) is DKK 4800.00 for kilolitre (the *BaseQuantity*). Because the supplier sells the oil in barrels of 750 litres and not kilolitres, the supplier must specify the conversion factor (*OrderableUnitFactorRate*) that should be applied to convert the supplier's base quantity to the unit of 1 barrel. In this case, the *OrderableUnitFactorRate* is 0.75. (1000 litres * 0.75 = 750 litres ≈ 1 barrel).

The price of one barrel of oil is calculated by multiplying the supplier's base price (*PriceAmount*) with the *OrderableUnitFactorRate*, that is DKK 4800.00 * 0.75 or DKK 3600.00 per barrel.

5. Relevant code lists

Code list:	Agency:	Urn:	Example value:
CurrencyCode	6	ISO 4217 Alpha	DKK, EUR
UnitOfMeasureCode	6	UN/ECE rec 20	PK, EA

6. Terms and abbreviations

Listed below are the most important terms and abbreviations:

Term:	Explanation:
Document level	Elements at document level are found directly under the root element (the top element) in the XML structure. elements at the document level apply to the whole document.
Line level	Elements at line level, unlike elements at the document level, only apply to a specific transaction line
Class	A class is a collection of elements. For example, the Price class contains elements such as PriceAmount, BaseQuantity, etc.
Element	An element is an information entity in an XML structure. For example, the PriceAmount is the element containing the price in an invoice line.
Attributes	In an XML element, it is possible to specify a property as an attribute, e. g. the attribute unitCode in which the unit for a quantity may be specified, as in the example: <cbc:BaseQuantity unitCode="BO">1</cbc:BaseQuantity>