

The UBL Mapping Committee have undertaken a review of xCBL 3.0 with respect to the semantic registry that has been developed by the UN/CEFACT Core Component initiative. This document outlines the recommended approach for mapping xCBL structures to corresponding Core Component construct and vice versa (where they exist). Equally importantly, we propose methodology will assist in the ongoing identification of changes to xCBL and candidate Core Component definitions as are necessary to support a workable mapping.

The approach taken by the Mapping Team is based on a few assumptions on the emerging work of Core Components. These are:

- "Core Components" are intended to present a set of "generic" business semantics that function like abstract classes.
- A Business Information Entity is a component used in context.
- In the cases when a given structure is useful exactly as presented, it inherits the properties of its core component directly in becoming a BIE. This is very often the case with simple, data-containing structures, although they may be renamed in some cases, provided that the re-naming affects the semantic of the component. Typically, this would be a case of semantic qualification, where a "basic" (that is, simple) Business Information Entity is a subset of the core component definition.

The relationship between these concepts and xCBL are:

- Core Component (semantic model) + Context = Business Information Entity (semantic model).
- Business Information Entities (semantic models, refined) + specific syntax binding = specific business vocabulary in XML (e.g. xCBL 3.0).

Therefore, it can be said that xCBL (and any other business vocabulary) is a syntax-bound, in-context use of a set of core components.

Whilst, in mapping xCBL, we want as much context-specific richness as we can get, our initial focus will be to map the "80%" documents for a range of common processes.

Based on these assumptions, the Team wish to propose a mapping methodology as follows.

This process description is accompanied by a worked example using the xCBL elements UnitPrice, Price and its construct as BuyerExpectedUnitPrice, which is expressed as:

```
<BuyerExpectedUnitPrice>  
  <Price>  
    <UnitPrice>00000000010.0000</UnitPrice>  
  </Price>  
</BuyerExpectedUnitPrice>
```

- (A) Start by mapping every simple data-level core component as well as any obvious higher-level constructs of the current CC library to what is in xCBL. The expectation is that that this will achieve somewhere between 20-50% alignment. Significantly, this will also identify the existence of "context-specific" structures ear-marked for the next stage of the process.

The core component, unit charge price.amount appears comparable to the xCBL element, UnitPrice. Note, that they are not exact matches, but both carry a number and a currency code. What we are seeking at this level is a broad-brush alignment.

NB.

It should be noted that some constructs in xCBL are based on assumptions about their use (e.g., assumed context drivers like Business Process and Industry). As our initial aim is to provide a small set of generic documents, we should avoid these constructs until Phase II of UBL. Therefore, at the end of this task, we will decide on what is to be kept, and what xCBL constructs may be deferred, before moving onto step B.

- (B) Using the mappings from step A., and the existing xCBL library, we reverse-engineer the process, to:
- a. Look for "in-context" matches between mapped items found in step A and establish the Business Information Entity and contexts involved.

The xCBL UnitPrice construct appears in several xCBL documents, for example within Order there is a construct called BuyerExpectedUnitPrice.

By taking the Core Component, unit charge price.amount and applying the context drivers, Business Process = "Place Order" and Role = "Buyer", we create a Business Information Entity for the pricing information for a basic unit of an item. It is this "in-context" definition that maps to the xCBL, BuyerExpectedUnitPrice.

- b. Document syntax-binding rules to help in deciphering the rest of the Core Component library.

The Core Component, unit charge price.amount is of the Core Component Type, amount.type. This implies a relationship between amount.types and other instances of the xCBL element Price wherever they may appear.

- (C) Identify the xCBL structures that aren't yet mapped, and determine:

- a. Which of these constructs can usefully be assembled from lower-level core components that exist.

Are there any more amount.type components used in xCBL structures in a different context?

- b. Any structures that may be absent from the CC library. It should be noted that the CC library is a work-in-progress and these missing items should be submitted as candidate core components to expand the library.

xCBL UnitPrice also contains, UnitOfMeasurement, the unit of measurement that the unit price is based on.

- c. Any inconsistencies in the xCBL library. We should not assume that xCBL definitions consistently follow their own constructs. These items should be identified for modification.

The Core Component, charge.price is also apparently related to the xCBL element, ProductPrice. Unfortunately, ProductPrice does not utilise the xCBL Price element and chooses to redefine certain elements, creating duplicate definitions (e.g. UOM rather than UnitOfMeasurement). Both these xCBL elements specify the units that relate to the measurement or quantity.

- (D) Produce a map that has xCBL elements, sets of context drivers and their values, and the Core Component(s) involved.

The syntax-binding that produces BuyerExpectedUnitPrice as it exists in xCBL and Core Components.

XCBL construct	Mapped xCBL element	Context 1 name/value	Context 2 name/value	Core Component
BuyerExpectedUnitPrice	UnitPrice <ul style="list-style-type: none"> • UnitPriceValue • Currency • UnitOfMeasurement 	Business Process = Place Order	Role = Buyer	unit charge price.amount (000146)
	Price			amount.type(000105) <ul style="list-style-type: none"> • amount. content (000106) • amount currency. identification. code (000107)

As a validation of this methodology, we propose that steps B, C and D be initially prototyped using a constrained set of xCBL structures (for example, the xCBL construct, ItemDetail). This would be a work item for UBL workshops in the week commencing October 29th 2001.