NES

Information Model Architecture

Version 2.0

NES Information Model Architecture

NES Information Model Architecture

---

1   introduction

The Northern European Subset (NES) group was established to enable interoperability of procurement data between users of the Universal Business Language (UBL). UBL is a royalty-free library of XML documents addressing the requirements of electronic procurement and international trade and transportation. Its second version (UBL 2.0) was released as an OASIS standard in December 2006. NES members contributed extensively to the development of this version of the standard.

The focus of NES is to define the specific use of UBL 2.0 electronic procurement documents domestically and between the member countries. The definition covers semantic interoperability within and between all business sectors, public and private.

This document describes the Information Model Architecture used by NES.

Note that there are several strategies regarding how to create, maintain, explain, use and re-use a UBL subset. The UBL Technical Committee is currently developing a guide on how to customize UBL documents; the NES group contributes to this guide and aligns where necessary.

2   objectives

The objectives of an Information Model Architecture are several:

consistency and robustness	the model should define and enforce business rules
ease of maintenance	corrections in documentation should only be actioned once, with automatic propagation of the change throughout the model. It must be possible to upgrade easily to a new version of the base standard
ease of explanation	it must be possible to communicate the model to audiences with little experience in information modelling
requirement for artefacts	it must be defined which artefacts are generated and how and where should they be deployed
auto-generation of artefacts	schemas, scripts and documentation must be automatically generated from the model
normative elements	normative elements of the Information Model and the artefacts must be defined

3   definition of terms

Definitions from UN/CEFACT Core Components Technical Specification 2.01 and Wikipedia.

BIE	a piece of business data or a group of pieces of business data with a unique Business Semantic definition. A Business Information Entity can be a Basic Business Information Entity (BBIE), an Association Business Information Entity (ASBIE), or an Aggregate Business Information Entity (ABIE).
BBIE	a Basic Business Information Entity that represents a singular business characteristic of a specific Object Class in a specific Business Context. A BBIE has a unique Business Semantic definition. A BBIE represents a Basic Business Information Entity Property and is therefore linked to a Data Type, which describes it values. A BBIE is derived from a Basic Core Component (BCC).
ASBIE	a Business Information Entity that represents a complex business characteristic of a specific Object Class in a specific Business Context. An Association Business Information Entity (ASBIE) has a unique Business Semantic definition. An ASBIE represents an Association Business Information Entity Property and is associated to an Aggregate Business Information Entity (ABIE), which describes its structure. An ASBIE is derived from an Association Core Component (ASCC).
ABIE	a collection of related pieces of business information that together convey a distinct business meaning in a specific Business Context. Expressed in modelling terms, an Aggregate Business Information Entity (ABIE) is the representation of an Object Class in a specific Business Context.
business context	the formal description of a specific business circumstance as identified by the values of a set of Context Categories, allowing different business circumstances to be uniquely distinguished.
XSD	an XML Schema Definition (XSD) is an instance of an XML schema written in the XML Schema language. An XSD defines a type of XML document in terms of constraints upon what elements and attributes may appear, their relationship to each other, what types of data may be in them etc. XSD can be used with validation software in order to ascertain whether a particular XML document is of a particular type, and to produce a Post-Schema Validation Infoset.

Schematron   Schematron is an XML structure validation language using patterns in treea tree structure. It is a simple and powerful structural schema language.

namespace   An XML Namespace is a W3C standard for providing uniquely named elements and attributes in an XML instance. An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace, then the ambiguity between identically named elements or attributes can be resolved.

4   conformance

This document discusses conformance with respect to business document instances e.g. an XML-invoice, but the discussion applies equally to conformance in the context of business processes.

4.1   UBL conformance

UBL instance conformance is straightforward. An XML instance is considered UBL conformant if:
- there are no constraint violations when validating the instance against the published UBL schema
- there are no additions (except when using the extension-point)
- all mandatory elements are present
- values follow the rules of the data types

-

4.2   NES conformance

To be NES conformant, an instance must validate against the NES Generic Document business rules; this can be done with XML Schema and Schematron. A NES conformant instance is always also UBL conformant.

4.3   NES profile conformance

To be NES profile conformant, an instance must validate against the NES Profile business rules; this can be done using XML schema and Schematron. A NES profile conformant instance is always NES conformant and UBL conformant.

A UBL conformant instance *might* be conformant to NES, but a NES conformant instance is *always* conformant to UBL.

This hierarchic conformance rule is implemented through a simple, robust and consistent restriction methodology:

1. a refined schema can never extend a cardinality or data type
2. a refined schema can always further restrict cardinality and data type.
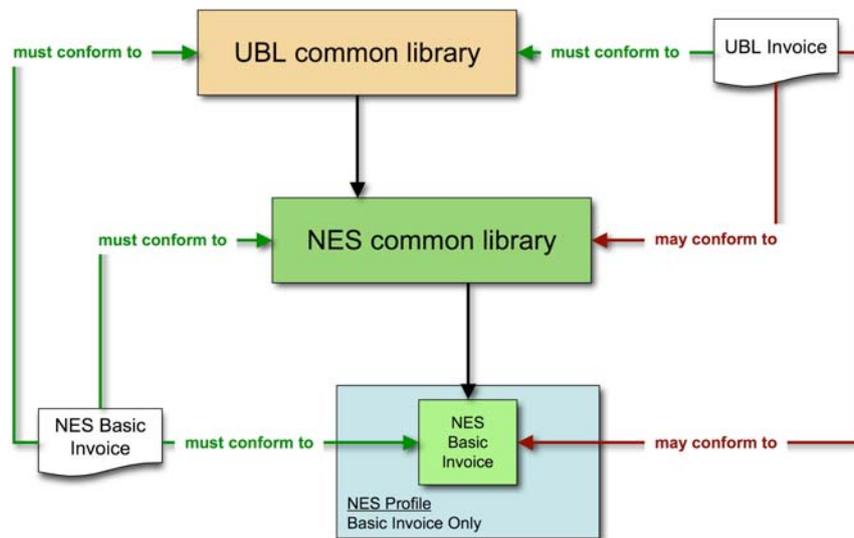
By keeping the UBL namespaces, all NES instances are always conformant to UBL software and validators.

5   the library approach

In UBL, all common aggregated components (ABIEs) such as Address, Delivery, Party, Payment Means etc. are stored in a library called the Common Library. The Common Library comprises approximately 113 reusable components. These components contain basic elements of information such as dates and identifiers; they also contain references to other components. The references between components makes the total model very comprehensive.

ABIEs are reused without regard to context. They can be reused by documents in Procurement and in Transportation; they can be reused within other ABIE's by association (ASBIE).

This, for example, means that an ABIE used in both Catalogue and Invoice is the same, regardless of the particular content requirements of the document in question. This results in a situation where some ABIEs cannot be used in a

straightforward and simple way.

The consequences of re-use are recognised and can be seen as negative consequence of the overall UBL approach. However, from the point of view of NES, the consistency benefits of re-use outweigh the detriments and NES now has a very rich Common Library that fulfils the requirements of all of the documents used.

5.1 NES libraries

The library-approach requires NES to create several levels of customized libraries, refined from the parent level library.
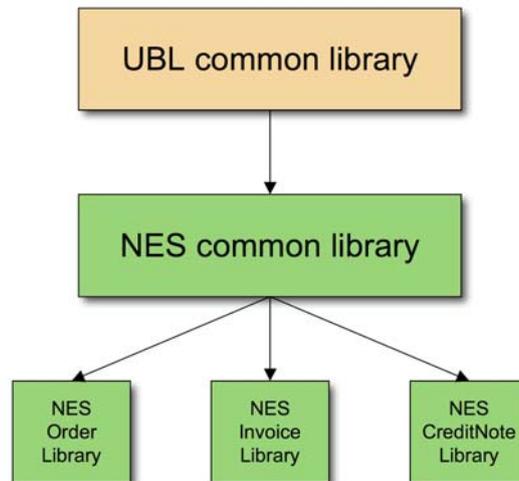
A lower level library cannot extend cardinality or add BIEs that are not part of the library directly above.

For example. the UBL Common Library ABIE "Party" has an BIE called Party Identifier which is unbounded; it can be repeated.

The NES Common Library also contains the "Party" ABIE, but restricts the Party Identifier so that it can be used only once.

Furthermore the NES Invoice Library can restrict the "Party" ABIE further, to the extent that the Party Identifier cannot be used at all, but it is not allowed to extend the cardinality to 'unbounded'.
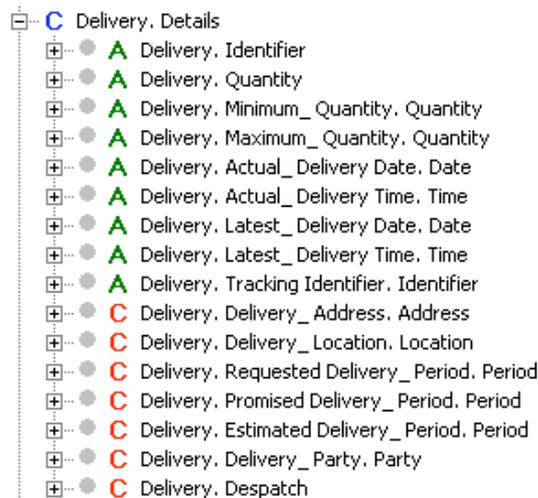
This one-way restriction enforces consistency and robustness of the Information Model Architecture.

5.2 illustrated restriction example

"Delivery" is one example of an ABIE that shares content (BIEs) across several documents and processes.

Below is how the "Delivery" ABIE is defined in the UBL Common Library. It comprises several BIEs that are not identified as required in the NES processes and documents.

```
☐ C Delivery. Details
   ⊞ ● A Delivery. Identifier
   ⊞ ● A Delivery. Quantity
   ⊞ ● A Delivery. Minimum_ Quantity. Quantity
   ⊞ ● A Delivery. Maximum_ Quantity. Quantity
   ⊞ ● A Delivery. Actual_ Delivery Date. Date
   ⊞ ● A Delivery. Actual_ Delivery Time. Time
   ⊞ ● A Delivery. Latest_ Delivery Date. Date
   ⊞ ● A Delivery. Latest_ Delivery Time. Time
   ⊞ ● A Delivery. Tracking Identifier. Identifier
   ⊞ ● C Delivery. Delivery_ Address. Address
   ⊞ ● C Delivery. Delivery_ Location. Location
   ⊞ ● C Delivery. Requested Delivery_ Period. Period
   ⊞ ● C Delivery. Promised Delivery_ Period. Period
   ⊞ ● C Delivery. Estimated Delivery_ Period. Period
   ⊞ ● C Delivery. Delivery_ Party. Party
   ⊞ ● C Delivery. Despatch
```

The "Delivery" ABIE is restricted to meet NES requirements at NES Common Library level (see below).

```
☐ C Delivery. Details
   ⊞ ● A Delivery. Identifier
   ⊞ ● A Delivery. Quantity
   ⊞ ● X Delivery. Minimum_ Quantity. Quantity
   ⊞ ● X Delivery. Maximum_ Quantity. Quantity
   ⊞ ● A Delivery. Actual_ Delivery Date. Date
   ⊞ ● A Delivery. Actual_ Delivery Time. Time
   ⊞ ● A Delivery. Latest_ Delivery Date. Date
   ⊞ ● A Delivery. Latest_ Delivery Time. Time
   ⊞ ● X Delivery. Tracking Identifier. Identifier
   ⊞ ● X Delivery. Delivery_ Address. Address
   ⊞ ● C Delivery. Delivery_ Location. Location
   ⊞ ● C Delivery. Requested Delivery_ Period. Period
   ⊞ ● C Delivery. Promised Delivery_ Period. Period
   ⊞ ● C Delivery. Estimated Delivery_ Period. Period
   ⊞ ● C Delivery. Delivery_ Party. Party
   ⊞ ● X Delivery. Despatch
```

However, even at NES Common Library level, the restricted "Delivery" ABIE still contains several BIEs that do not make sense in the context of, for example, an Invoice e.g. Minumum_Quantity and Latest_DeliveryDate; these BIE's are used in "Delivery" in the Order.

Therefore, NES requires one more level of refinement (restriction) where only BIEs relevant to the Invoice are present; the NES Invoice Library restriction is shown below.

6   internal structure of the models

NES restrictions are collected on several levels:

1. the UBL libraries and documents where nothing is restricted
2. the NES Common Library that includes everything that can be used in all the NES documents
3. the NES document specific libraries and generic documents comprising only BIEs relevant for specific documents.

The final level is the one at which Profiles are defined. Using the generic documents as base, further restrictions are applied to define process specific (Profile) documents.