The Future of XML Vocabularies

OASIS SYMPOSIUM

24 April: Tutorials
8:30 AM – 12:00 PM
New Orleans Marriott

Creating UBL Conformant Schema Tutorial

Mark Crawford
Senior Research Fellow
LMI Government Consulting
mcrawford@lmi.org
Agenda

- UBL Overview
- The Modeling Methodology
- Core Components and Business Information Entities
- XML NDR
- Creating the Schema
- Customizing UBL
Why UBL?

- ebXML Core Components are "syntax neutral", it will be a basis for multiple business document dialects and standards

- But we must have concrete standard XML syntax to enable wide use and cheap commercial software

- Given a concrete XML syntax for business, users will adopt it

UBL is developing XML business document design rules, XML syntax based on ebXML core component (CC) structures and ebXML (UN/CEFACT) CC compliant XML document schemas
The UBL Development Approach

component model

Manual assembly of document structures as BIEs

spreadsheet assembly model

Automated transformation according to UBL Naming and Design Rules

implementation model

Automated transformation

schemas
The Interoperability Challenge

The Virtual Enterprise

Shipping Note

- Customer Reference: HSGF1220
- Goods Description: 1 of Document Engineering, Glushko & McGrath
- Shipping Address: New Bedford, Rhode Island, USA

GMBooks.com

Purchase Order

- Order Reference: HSGF1220
- Item Ordered: Document Engineering, Glushko & McGrath
- Quantity Required: 1
- Shipping Address: New Bedford, Rhode Island, USA

Credit Authority

- Delivery Service

Transaction Advice

- Reference: 99847566663
- Details: HSGF-1220 GMBooks.com
- Payment Method: Viza
- Billing Address: Fremantle, Western Australia
- Grand Total: 105.15

Customer’s View of Buying a Book

*Taken from: Document Engineering, Glushko and McGrath, MIT Press, 2005*
Understanding Documents

• Interoperability means understanding the meaning of documents and their information components.
• This is facilitated when their **semantics**, **structure** and **syntax** conform to standard patterns.
• XML has become the preferred **syntax** pattern for representing information in documents.
• Now we need to define common patterns for the **semantics** and **structure** of business documents.
Document Engineering

- A new discipline needed for analyzing and designing new business documents.
- Synthesizes complementary ideas from business analysis, task analysis, document analysis and data analysis.
- The OASIS UBL TC has document engineered re-usable semantic and structural patterns for common business requirements…
- … to create a **Universal Business Language**.
Creating Conceptual Component Models

Business Operations View

UML and spreadsheets

Conceptual models showing all possible associations

Functional Service view

Schemas

The Real World

Messages/Documents

Analysis

XML Schema Libraries

Database Schemas

EDI Message Definitions

Business Document Models

Limited interoperability
Component Model for “Item”
Creating the Document Structures

Business Operations View

UML and spreadsheets
Design

Functional Service view

XML Schema Libraries
Database Schemas
EDI Message Definitions
Business Document Models

Documents structures are assembled from 'network' of components into document models

Limited interoperability
## A Spreadsheet Sample - Item

<table>
<thead>
<tr>
<th>UBL Name</th>
<th>BIE Dictionary Entry Name</th>
<th>Occurrence Type</th>
<th>BIE Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Item. Description. Text</td>
<td>0..1</td>
<td>BBIE a free form field that can be used to give a text</td>
</tr>
<tr>
<td>PackQuantity</td>
<td>Item. Pack. Quantity</td>
<td>0..1</td>
<td>BBIE the unit packaging quantity.</td>
</tr>
<tr>
<td>PackSizeQuantity</td>
<td>Item. Pack_Size. Quantity</td>
<td>0..1</td>
<td>BBIE the number of items in a pack.</td>
</tr>
<tr>
<td>FromCatalogueIndicator</td>
<td>Item. From Catalogue. Indicator</td>
<td>0..1</td>
<td>BBIE an indicator that denotes whether or not the item was</td>
</tr>
<tr>
<td>BuyersItemIdentification</td>
<td>Item. Buyers_ Item Identification</td>
<td>0..1</td>
<td>ASBI associates the item with its identification according to the buyers system.</td>
</tr>
<tr>
<td>SellerItemIdentification</td>
<td>Item. Sellers_ Item Identification</td>
<td>0..1</td>
<td>ASBI associates the item with its identification according to the sellers system.</td>
</tr>
<tr>
<td>ManufacturersItemIdentification</td>
<td>Item. Manufacturers_ Item Identification</td>
<td>0..1</td>
<td>ASBI associates the item with its identification according to the manufacturers system.</td>
</tr>
<tr>
<td>StandardItemIdentification</td>
<td>Item. Standard_ Item Identification</td>
<td>0..1</td>
<td>ASBI associates the item with its identification according to a standard system.</td>
</tr>
<tr>
<td>CatalogueItemIdentification</td>
<td>Item. Catalogue_ Item Identification</td>
<td>0..1</td>
<td>ASBI associates the item with its identification according to a cataloging system.</td>
</tr>
<tr>
<td>AdditionalItemIdentification</td>
<td>Item. Additional_ Item Identification</td>
<td>0..n</td>
<td>ASBI associates the item with other identification means</td>
</tr>
<tr>
<td>CatalogueReference</td>
<td>Item. Catalogue_ Reference</td>
<td>0..1</td>
<td>ASBI associates the item with the catalogue from which the item was selected.</td>
</tr>
<tr>
<td>OriginCountry</td>
<td>Item. Origin_ Country</td>
<td>0..1</td>
<td>ASBI associates the item with its country of origin</td>
</tr>
<tr>
<td>CommodityClassification</td>
<td>Item. Commodity Classification</td>
<td>0..1</td>
<td>ASBI associates the item with its classification(s) according to a commodity classifying system.</td>
</tr>
<tr>
<td>SalesConditions</td>
<td>Item. Sales. Conditions</td>
<td>0..n</td>
<td>ASBI associates the item with sales conditions appertaining to it</td>
</tr>
<tr>
<td>HazardousItem</td>
<td>Item. Hazardous Item</td>
<td>0..n</td>
<td>ASBI associates the item with its hazardous item information.</td>
</tr>
<tr>
<td>TaxCategory</td>
<td>Item. Tax Category</td>
<td>0..n</td>
<td>ASBI associates the item with one or more taxes</td>
</tr>
<tr>
<td>BasePrice</td>
<td>Item. Base. Price</td>
<td>0..n</td>
<td>ASBI associates the item with one or more base prices.</td>
</tr>
</tbody>
</table>
The Core Components Technical Specification

ISO/TS 15000-5:2004

electronic business Extensible Markup Language (ebXML) -- Part 5: Core Components Technical Specification (ebCCTS)
ISO 15000-5 The Core Components
Technical Specification (CCTS)

- Implementation rules for ISO 11179 parts 4 and 5
- A methodology for developing a common set of semantic building blocks representing general types of business data
  - Adds structure and consistency to database constructs
  - Provides for the creation of new business vocabularies and restructuring of existing business vocabularies
  - Is flexible and interoperable
- Defines a syntax-neutral meta-model for business semantics (meaning of words)

[Diagram depicting Address, Organization, Line Item, and Purchase Order]
ISO 11179 Data Constructs

Data Model

Entity (Type)

Attribute (Type)

Data Element Classification Structure

Object Class

Property

Representation

Data Element Concept

Data Element (Type)

Generic Data Element
The Baseline – ISO 15000-5 Follows ISO 11179

- This is basic object-oriented “good stuff”

### Object class

_____________________
Property 1: representation 1
Property 2: representation 2
Property 3: representation 3
Property 4: representation 4

### Address

_____________________
Street: text
Post code: text
Town: text
Country: identifier

ISO 11179 governs data dictionaries: defines the notions of object class, property, and representation term
Key Concepts

• ISO 15000-5 provides the semantic, syntactic, lexical, and uniqueness rules called for in ISO 11179-5

• Approach is more flexible than current standards in this area because the semantic standardization is done in a syntax-neutral fashion

• Two trading partners using different syntaxes [e.g. XML and EDI are using Business Semantics in the same way]

• Common Core Components underpinnings enable clean mapping between disparate databases and message definitions across syntaxes, industry, and regional boundaries
Core Component Overview

Four categories of Core Components:

- Core Component Type (CCT)
- Basic Core Component (BCC)
- Aggregate Core Component (ACC)
- Association Core Component (ASCC)

FIGURE 5.2: Core Component Hierarchies

<table>
<thead>
<tr>
<th>Supplementary Component</th>
<th>Core Component Type (CCT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consists of</td>
<td>Consists of</td>
</tr>
<tr>
<td>Content Component</td>
<td>Specifies restrictions on</td>
</tr>
<tr>
<td>Data Type</td>
<td>Defines set of values of</td>
</tr>
<tr>
<td>Basic Core Component</td>
<td>Basic Core Component (BCC)</td>
</tr>
<tr>
<td></td>
<td>Provides a simple characteristic of and is aggregated in</td>
</tr>
<tr>
<td>Association Core</td>
<td>Aggregate Core Component (ACC)</td>
</tr>
<tr>
<td>Component (ASCC)</td>
<td>Provides a complex characteristic of and is aggregated in</td>
</tr>
</tbody>
</table>
Technical Details CC and Data Types Metamodel
Core Component (CC) Definition

- A building block for the creation of a semantically correct and meaningful information exchange package
- Known as Core Components (CCs)
- Contains only the information pieces necessary to describe a specific concept
- Basis to construct all electronic business messages
- Basis for Business Information Entities

**Invoice**

```
Address. City Name. Text
```

```
Address. Line One. Text
```

```
Address. Details
```

---

©Mark Crawford, LMI Government Consulting, April 2005
Core Component Type (CCT) Definition

- A Core Component, which consists of the actual data content plus one or more Supplementary Components that give essential extra definition to the Content Component.
- Does not have Business Semantics.
- Example: CCT for a specific amount of currency:
  - Amount.Type
Content Component and Supplementary Component

- Core Component Type = 12 Euro

- Content Component – defines the Primitive Type used to express the content of a Core Component Type
  - Example content component or Data value: 12
  - This content component has no semantic meaning on its own

- Supplementary Component – gives additional meaning to the Content Component in the Core Component Type
  - Example: Euro
  - Gives the essential extra definition/semantic meaning to the content component
Core Component Type (CCT) Example

Core Component Type: \textbf{Measure. Type}

Content Component: \textbf{15.45} \textit{Value} has no semantic meaning

Supplementary Component: \textbf{Inches} \textit{Essential extra definition}

\textbf{15.45 Inches}
Core Component Types

- [C7] The Core Component Type shall be one of the approved Core Component Types
  - The approved core component types are contained in Table 8-1 in ISO 15000-5

- Amount. Type
- Binary Object. Type
- Code. Type
- Date Time. Type
- Identifier. Type
- Indicator. Type
- Measure. Type
- Numeric. Type
- Quantity. Type
- Text. Type
Core Component Type

- [C8] The Content Component shall be the approved Content Component for the related Core Component Type
  - The approved content component types are contained in Table 8-2 in ISO 15000-5

- Identifier. Content
- Code. Content

- [C9] The Supplementary Component shall be one of the approved Supplementary Components for the related Core Component Type
  - The approved supplementary component types are contained in Table 8-2 in ISO 15000-5

- Code List. Agency. Identifier
- Date Time. Format. Text
- Identification Scheme. Version. Identifier
- Measure Unit. Code
6.1.2 Data Types

A Data Type defines the set of valid values that can be used for a particular Basic Core Component Property or Basic Business Information Entity Property. It is defined by specifying restrictions on the Core Component Type from which the Data Type is derived. Figure 6-1 describes the Data Type and shows relationships to the Core Component Type.
CCTS Data Type Rule #1

[D1] A Data Type shall be based on one of the approved Core Component Types.

- Some Core Component Types have more than one representation term (See table 8-3 in CCTS)
- This means that there are more data types than core component types
# List of Permissible Representation Terms

<table>
<thead>
<tr>
<th>Representation Term</th>
<th>Related CCT</th>
<th>Secondary Rep Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Amount. Type</td>
<td></td>
</tr>
<tr>
<td>Binary Object</td>
<td>Binary Object. Type</td>
<td>Graphic, Picture, Sound, Video</td>
</tr>
<tr>
<td>Code</td>
<td>Code. Type</td>
<td></td>
</tr>
<tr>
<td>Date Time</td>
<td>Date Time. Type</td>
<td>Date, Time</td>
</tr>
<tr>
<td>Identifier</td>
<td>Identifier. Type</td>
<td></td>
</tr>
<tr>
<td>Indicator</td>
<td>Indicator. Type</td>
<td></td>
</tr>
<tr>
<td>Measure</td>
<td>Measure. Type</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>Numeric. Type</td>
<td>Value, Rate, Percent</td>
</tr>
<tr>
<td>Quantity</td>
<td>Quantity. Type</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Text. Type</td>
<td>Name</td>
</tr>
</tbody>
</table>

Table 8-3 CCTS V1.9

Core Components Technical Specification V2.01
Part 8 of the ebXML Framework
CCTS Data Types Rule #2

[D2] Where necessary, a Data Type shall restrict the set of valid values allowed by the Core Component Type on which it is based, by imposing restrictions on the Content Component and/or the Supplementary Component.

With known business semantics

- Supplementary Component
- Content Component
- Rep. Term (Data Type)
- Core Component Type (CCT)
- Association Core Component (ASCC)
- Basic Core Component (BCC)
- Aggregate Core Component (ACC)

Without business semantics

Provides a simple characteristic of and is aggregated in

Provides a complex characteristic of and is aggregated in

Consists of

Consists of

Specifies restrictions on

Defines set of values of
Summary: Core Component Constructs

- **Basic Core Component (BCC)**
- **Core Component Type (CCT)**
- **Association Core Component (ASCC)**
- **Aggregate Core Component (ACC)**

**Without business semantics**
- **Content Component**
  - Consists of
  - Specifies restrictions on

**With known business semantics**
- **Data Type**
  - Consists of
  - Defines set of values of
- **Supplementary Component**
  - 1-n
- **Basic Core Component (BCC)**
  - Provides a simple characteristic of and is aggregated in
- **Aggregate Core Component (ACC)**
  - Provides a complex characteristic of and is aggregated in

©Mark Crawford, LMI Government Consulting, April 2005
Basic Core Component (BCC) Definition

- A Core Component which constitutes a singular business characteristic of a specific Aggregate Core Component that represents an Object Class
- It has a unique Business Semantic definition
- Represents a Basic Core Component Property and is therefore of a Data Type, which defines its set of values
- Function as the Properties of Aggregate Core Components
Basic Core Component Example

- Item. Name. Text
- Organization. Name. Text
- Organization. Description. Text
- Address. Street. Text
- Address. City Name. Text
Basic Core Component (BCC)

- Property of an Aggregate Core Component
  - Attribute of a class
Basic Core Component (BCC) Example

- Has a Basic Core Component Property

Aggregate Core Component

Basic Core Components

<table>
<thead>
<tr>
<th>Address. Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address. Line One. Text</td>
</tr>
<tr>
<td>Address. Line Two. Text</td>
</tr>
<tr>
<td>Address. Postcode. Code</td>
</tr>
</tbody>
</table>

Basic Core Component Property

- Line One. Text
- Line Two. Text
- Postcode. Code
Basic Core Component (BCC) Example

- Has a Data Type

Aggregate Core Component

Basic Core Components

<table>
<thead>
<tr>
<th>Address. Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address. Line One. Text</td>
</tr>
<tr>
<td>Address. Line Two. Text</td>
</tr>
<tr>
<td>Address. Postcode. Code</td>
</tr>
</tbody>
</table>

Data Type

- Address._Text. Type
- Address._Code. Type
Aggregate Core Component (ACC) Definition

- A collection of related pieces of business information that together convey a distinct business meaning
- Independent of any specific Business Context
- Expressed in modeling terms, it is the representation of an Object Class
Aggregate Core Component (ACC)

- Representation of an Object Class
- In a real business circumstance serves as the basis of an Aggregate Business Information Entity

Class Diagram

<table>
<thead>
<tr>
<th>Class Attributes</th>
<th>Class</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core Components

<table>
<thead>
<tr>
<th>Aggregate Core Component</th>
<th>Item. Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item. Name. Text</td>
<td></td>
</tr>
<tr>
<td>Item. Quantity</td>
<td></td>
</tr>
<tr>
<td>Item. Identification. Identifier</td>
<td></td>
</tr>
</tbody>
</table>
Aggregate Core Component (ACC) Rules

• [C2] Within an Aggregate Core Component, all embedded Core Component Properties shall be related to the concept of the aggregate property
  – Example:

<table>
<thead>
<tr>
<th>Aggregate Core Component</th>
<th>Address. Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Core Components</td>
<td>Address. Line One. Text</td>
</tr>
<tr>
<td></td>
<td>Address. Line Two. Text</td>
</tr>
<tr>
<td></td>
<td>Address. Postcode. Code</td>
</tr>
</tbody>
</table>

All Properties are a characteristic of an address
Aggregate Core Component (ACC)

- Has a Core Component Property that defines the business characteristic

Aggregate Core Component

Basic Core Components

Person. Details

Person. Name. Text
Person. Gender. Indicator
Person. Title. Text

Property

Person
Aggregate Core Component (ACC) Rules

- [C3] There shall be no semantic overlap between the Core Component Properties embedded within the same Aggregate Core Component
  - Example:

<table>
<thead>
<tr>
<th>Aggregate Core Component</th>
<th>Address. Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Core Components</td>
<td>Address. Line One. Text</td>
</tr>
<tr>
<td></td>
<td>Address. Line Two. Text</td>
</tr>
<tr>
<td></td>
<td>Address. Postcode. Code</td>
</tr>
</tbody>
</table>

Each basic property has a unique semantic meaning as part of an address
Aggregate Core Component (ACC) Rules

• [C5] An Aggregate Core Component shall contain at least one Core Component Property. A Core Component Property shall be either a Basic Core Component Property or an Association Core Component Property.
  – Example:

<table>
<thead>
<tr>
<th>Core Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate Core Component</td>
</tr>
<tr>
<td>Basic Core Components</td>
</tr>
<tr>
<td>Person. Details</td>
</tr>
<tr>
<td>Person. Name. Text</td>
</tr>
<tr>
<td>Person. Gender. Indicator</td>
</tr>
<tr>
<td>Person. Title. Text</td>
</tr>
<tr>
<td>Person. Address</td>
</tr>
<tr>
<td>Association Core Component</td>
</tr>
</tbody>
</table>

One property must be present, either as a BCC or an ASCC.
Aggregate Core Component (ACC)
Examples

- Contact. Details
- Delivery. Details
- Facility. Details
- Location. Details
- Organization. Details
- Party. Details
- Report. Details
Association Core Component (ASCC) Definition

- A Core Component which constitutes a complex business characteristic of a specific Aggregate Core Component that represents an Object Class.
- It has a unique Business Semantic definition.
- Represents an Association Core Component Property and is associated to an Aggregate Core Component, which describes its structure.
Association Core Component (ASCC)

- An ASCC is a Core Component naming mechanism for expressing the relationship between two object classes
  - Object Oriented inheritance that retains semantic clarity that cannot be expressed in UMM
  - Expresses the structure of the association

Class Diagram

- Organization
  - Name
  - Identification
- Address
  - Line One
  - Line Two
  - City Name
  - Postal Code

Association

Aggregate Core Component

- Organization
  - Name
  - Identification

Contact

Aggregate Core Component

- Address
  - Line One
  - Line Two
  - City Name
  - Postcode
Association Core Component (ASCC) Example

ASCC =

- Object Class Term of the ACC that contains the ASCC (Person)
- Property Term that represents the property of the ASCC (Official/Residence)
- Object Class Term of the ACC that describes the structure of the ASCC (Address)

Core Components
- Person. Residence. Address (ASCC)
- Person. Official. Address (ASCC)
Association Core Component (ASCC)

- Expressing the structure of the association

The structure of the Association Core Component is described by Address. Details
**Aggregate Core Component (ACC) Nested Association Rule**

- [C6] An Aggregate Core Component shall never contain – indirectly or at any nested level – a mandatory Association Core Component Property that references itself.
  - Example:

<table>
<thead>
<tr>
<th>Core Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate Core Component</td>
</tr>
<tr>
<td>![Arrow to Facility. Details]</td>
</tr>
<tr>
<td>Basic Core Components</td>
</tr>
<tr>
<td>![Arrow to Name. Text]</td>
</tr>
<tr>
<td>![Arrow to Type. Code]</td>
</tr>
<tr>
<td>![Arrow to Identification. Identifier]</td>
</tr>
<tr>
<td>Association Core Component</td>
</tr>
<tr>
<td>![Arrow to Facility. Facility]</td>
</tr>
</tbody>
</table>

A mandatory ASCC would cause looping.

Not mandatory.
The Rest of the Story

- Core Components are the building blocks for Business Information Entities.
- The key differentiator between Core Components and Business Information Entities is the concept of Business Context.
- Business Context is a mechanism for qualifying and refining Core Components according to their use under particular business circumstances.
- Once Business Contexts are identified, Core Components can be differentiated to take into account any necessary qualification and refinement needed to support the use of the Core Component in the given Business Context.
Business Context Definition

- The formal description of a specific business circumstance as identified by the values of a set of Context Categories
  - Allows different business circumstances to be uniquely distinguished
- ISO 15000-5 identifies eight context categories
  - Business Process, Production Classification, Industry Classification, Geopolitical, Official Constraints, Business Process Role, Supporting Role, and System Capabilities

Example: Geopolitical Contexts – allow description of those aspects related to region, nationality, or geographically based cultural factors.

  Global, Continent, Economic Region, Country
Context Application

- A set of eight values identifies a unique business context
  - Business Process
  - Product Classification
  - Industry Classification
  - Geopolitical
  - Official Constraints
  - Business Process Role
  - Supporting Role
  - System Capabilities

![Diagram showing the relationship between core components, context categories, constraints, syntax neutral and syntax bound, and specific business context.]

- Syntax neutral
- Syntax bound

EDI MIG, XML DTD, XML Schema, others
Business Information Entity (BIE) Definition

- 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).
Business Information Entity (BIE)

- A Core Component used in a real business circumstance
- A Core Component with business context applied

**Core Component (CC)**

A building block for the exchange of semantically correct and meaningful information

**Apply business context**

**Business Information Entity (BIE)**

A Core Component to which a business context has been applied
Relationship of CCs and BIEs

- A Basic Business Information Entity is based on a Basic Core Component (BCC)
- An Aggregate Business Information Entity is a re-use of an Aggregate Core Component (ACC) in a specified Business Context
- An Association Business Information Entity is based on an Association Core Component (ASCC)
## CC/BIE Relationship Example

### Multiple ABIEs can be created from an ACC

<table>
<thead>
<tr>
<th>Core Components</th>
<th>Business Information Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Supplier. Organization. Department_ Name. Text (BBIE)</td>
</tr>
<tr>
<td>Address. Details (ACC)</td>
<td>Mailing. Address. Details (ABIE)</td>
</tr>
<tr>
<td></td>
<td>Shipping. Address. Details (ABIE)</td>
</tr>
</tbody>
</table>

### Multiple BBIEs can be created from a BCC
An ACC can be restricted

An ABIE does not need to include all attributes (BBIEs)

<table>
<thead>
<tr>
<th>Address. Details (ACC)</th>
<th>Mailing_ Address. Details (ABIE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address. Line One. Text (BCC)</td>
<td>Mailing_ Address. Line One. Text (BBIE)</td>
</tr>
<tr>
<td>Address. Line Two. Text (BCC)</td>
<td>Mailing_ Address. Line Two. Text (BBIE)</td>
</tr>
<tr>
<td>Address. City Name. Text (BCC)</td>
<td>Mailing_ Address. City Name. Text (BBIE)</td>
</tr>
</tbody>
</table>

BBIE is not included
Basic BIE (BBIE)

- Property of an Aggregate Business Information Entity
- Based on a Basic Core Component (BCC)

```
<table>
<thead>
<tr>
<th>Core Component</th>
<th>Aggregate Core Component</th>
<th>Basic Core Components</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Organization</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description Business Type</td>
</tr>
</tbody>
</table>
```

```
Aggregate Business Information Entity

Supplier_ Organization

Name

Description

Business Type

Basic Business Information Entities

Business Context

+ Supplier

= Aggregate Business Information Entity
```
Aggregate BIE (ABIE)

- Representation of an Object Class
- An Aggregate Business Information Entity is a re-use of an Aggregate Core Component (ACC) in a specified Business context
An ASBIE is a Business Information Entity naming mechanism for expressing the relationship between two Aggregate Business Information Entities for a specific instance.

This supplier organization has an official delivery address. Official is the property of the association.
Association BIE Example

The property expresses the relationship between the two ACCs

Business Information Entities
- US_Person. Residence. US_Address (ASBIE)
- US_Person. Official. US_Address (ASBIE)
Creating a Business Information Entity

Step 1: Determine Object Class
Step 2: Determine Property Term
Step 3: Determine Representation Term
Step 4: Determine Qualifier Term(s)
Step 5: Concatenate Name
Step 6: Complete Data Analysis Worksheet
Creating a Business Information Entity

Step 1: Determine Object Class

• Identify the logical grouping of elements
• Determine if this grouping has a singular business characteristic of an existing Aggregate Core Component in a specific Business Context
• Review controlled vocabulary of object class terms
Creating a Business Information Entity

Step 2: Determine Property Term

- Determine if the property distinguishes or describes the object class
- Determine if this property is a unique characteristic of the object class
- Review controlled vocabulary of property terms
Creating a Business Information Entity

Step 3: Determine Representation Term
- Determine the nature of the atomic data type that reflects the use of this construct
- Using the atomic data type, identify the appropriate permissible representation term from Table 8-3

Step 4: Determine Qualifiers
- Identify the business context for the component
- [B27] Qualifier Terms shall precede the associated Object Class Term or Property Term.

Inventory_ Organization. Department_ Name. Text
Creating a Business Information Entity

Step 5: Concatenate Name
- Comply with ISO 15000-5 Naming Rules
ISO 15000-5 ABIE DEN Rules

- [B30] The Dictionary Entry Name of an Aggregate Business Information Entity shall consist of the name of the Object Class of its associated Aggregate Core Component and possibly additional Qualifier Term(s) to represent its specific Business Context, followed by a dot, a space character, and the term "Details"

**Supplier_ Organization. Details**

- **Object Class Qualifier**
- **Object Class**
- **“Details”**
Aggregate BIE Example

Core Component

Aggregate Core Component

Basic Core Components

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Measure</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>

Aggregate Business Information Entity

Container_ Dimension

| Measure: Measure |
| Value: Measure |
| Type: Code |
| Description: Text |

Basic Business Information Entities

Container_ Dimension. Details (ABIE)

Qualifier Term

Object Class Term

Property Term
### Data Analysis Worksheet

#### Step 6: Complete Data Analysis worksheet

<table>
<thead>
<tr>
<th>Original Source Date Element Name</th>
<th>Definition</th>
<th>Dictionary Entry Name</th>
<th>ABIE/B BIE/AS BIE</th>
<th>Object Class Qualifier</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type</th>
<th>Associated Object Class Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The information relevant to a person or organization that acts as a point of contact with</td>
<td>Supplier_ Contact. Details</td>
<td>ABIE</td>
<td>Supplier</td>
<td>Contact</td>
<td>Details</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The position or designation of this contact person within an organization such as Director, Software Engineer, Purchasing Manager.</td>
<td>Supplier_ Contact. Job Title. Text</td>
<td>BBIE</td>
<td>Supplier</td>
<td>Contact</td>
<td>Job Title</td>
<td>Text Type</td>
<td>Text Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The textual description of any general or specific responsibilities related to this contact.</td>
<td>Supplier_ Contact. Responsibility. Text</td>
<td>BBIE</td>
<td>Supplier</td>
<td>Contact</td>
<td>Responsibility</td>
<td>Text Type</td>
<td>Text Type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tying It All Together

Core

- Core Component Type (CCT)
  - Specifies restrictions on

  - Data Type
    - Defines set of values of

      - Basic Core Component
        - As Property Aggregated in

        - Aggregate Core Component
          - As Property Aggregated in

          - Association Core Component
            - Qualifies the Object Class of

Business

- Business Information Entity
  - As Property Aggregated in

  - Aggregate Business Information Entity
    - Aggregated in
    - As Property Aggregated in

  - Aggregate Component
    - Adds extra information

  - Message Assembly

Core Component Library
A Reminder

- The term Core Component is used as a generic term that encompasses Basic Core Components, Association Core Components, Aggregate Core Components, and their associated Core Component Types.
- The term Business Information Entity is used as a generic term encompassing Basic Business Information Entities, Association Business Information Entities, and Aggregate Business Information Entities.
Technical Details –
Core Component Storage

• Section 7 fully describes storage requirements for all Core Component and Business Information Entity Constructs
• The rules consist of
  – Storing Core Components
  – Storing Data Types
  – Storing Context
  – Storing Business Information Entities
  – Core Component Storage Metadata
Creating the Schemas

Business Operations View
- UML and spreadsheets
- Design

Service View
- Schemas

The Real World
- Messages/Documents

Limited interoperability

XML Schema Libraries
- Database Schemas
- EDI Message Definitions
- Business Document Models

UBL script does this automatically

Implementation

Schema componentry based on UBL Naming and Design rules
<xsd:complexType name="ItemType">
<xsd:annotation>
<xsd:documentation>
<xcts:Component>
<xcts:CategoryCode>ABIE</xcts:CategoryCode>
<xcts:DictionaryEntryName>Item</xcts:DictionaryEntryName>
<xcts:Definition>Information directly relating to an item</xcts:Definition>
<xcts:ObjectClass>Item</xcts:ObjectClass>
<xcts:PropertyTerm>Details</xcts:PropertyTerm>
<xcts:RepresentationTerm>Details</xcts:RepresentationTerm>
<xcts:BusinessTerm>article,product,goods item</xcts:BusinessTerm>
<xcts:Component>
<xsd:documentation>
<xsd:annotation>
<xsd:sequence>
<xsd:element ref="Description" minOccurs="0"/>
...
...
...
The Standards Used

- **STA2** - All UBL schema and messages MUST be based on the W3C suite of technical specifications holding recommendation status.
Schema Structure

UBL Schema MUST conform to the following physical layout as applicable:

XML Declaration

<!-- ===== Copyright Notice ===== -->

<!-- ===== xsd:schema Element With Namespaces Declarations ===== -->

xsd:schema element to include version attribute and namespace declarations in the following order:

xmlns:xsd
Target namespace
Default namespace

CommonAggregateComponents
CommonBasicComponents
CoreComponentTypes
Unspecialized Datatypes
Specialized Datatypes
Identifier Schemes
Code Lists

Attribute Declarations – elementFormDefault="qualified" attributeFormDefault="unqualified"

<!-- ===== Imports ===== -->

CommonAggregateComponents schema module

CommonBasicComponents schema module

Unspecialized Types schema module

Specialized Types schema module

<!-- ===== Global Attributes ===== -->

Global Attributes and Attribute Groups

<!-- ===== Root Element ===== -->
Each UBL:DocumentSchema MUST identify one and only one global element declaration that defines the document ccts:Aggregate BusinessInformationEntity being conveyed in the Schema expression. That global element MUST include an xsd:annotation child element which MUST further contain an xsd:documentation child element that declares “This element MUST be conveyed as the root element in any instance document based on this Schema expression.”

Example:

```xml
<xsd:element name="Order" type="OrderType">
  <xsd:annotation>
    <xsd:documentation>This element MUST be conveyed as the root element in any instance document based on this Schema expression</xsd:documentation>
  </xsd:annotation>
</xsd:element>
```
[Definition] Document schema –

The overarching schema within a specific namespace that conveys the business document functionality of that namespace. The document schema declares a target namespace and is likely to pull in by including internal schema modules or importing external schema modules. Each namespace will have one, and only one, document schema.

- Document does not denote / connote a ‘narrative’ document
- These XML ‘document’ Schemas define XML transactions for exchange between app servers
Element Declarations

[ELD2] All element declarations MUST be global with the exception of ID and Code which MUST be local.

• Much discussion on this issue
• Ultimate deciders were:
  – desire to manage by both types and elements
  – XPath limitations

Late Breaking News - UBL 2.0 will be all Global
[NMS1] Every UBL-defined or -used schema module, except internal schema modules, MUST have a namespace declared using the xsd:targetNamespace attribute.

[NMS2] Every UBL-defined or -used schema set version MUST have its own unique namespace.

[Definition] Schema Set –
A collection of schema instances that together comprise the names in a specific UBL namespace.

[NMS3] UBL namespaces MUST only contain UBL developed schema modules.
Namespace URIs

- UBL has chosen URNs vice URLs as the Schema Location URI.
  - Primary differentiator is required run-time support and the need for persistence
  - Drawback is limit on URN resolvability
- RFC 2396 guides URI syntax
- RFC 3121 guides OASIS URN Namespace schemes

[NMS4] The namespace names for UBL Schemas holding committee draft status MUST be of the form:


[NMS5] The namespace names for UBL Schemas holding OASIS Standard status MUST be of the form:

Versioning

- UBL has decided to include versioning information as part of the document-id component of the namespace
- The version information is divided into major and minor fields
- The minor field has an optional revision extension
- For example
  - urn:oasis:names:specification:ubl:schema:xsd:Order-1.0
- A host of rules related to standardizing this
[Definition]
Internal schema module –
A schema that is part of a schema set within a specific namespace.

udt = Unspecialized Datatype, sdt = Specialized Datatype, cbc = Common Basic Components, cac = Common Aggregate Components, cct = Core Component Type
[SSM6]

All UBL internal schema modules MUST be in the same namespace as their corresponding document schema.
Limitations on Import

[SSM2] A document schema in one UBL namespace that is dependent upon type definitions or element declarations defined in another namespace MUST only import the document schema from that namespace.

[SSM3] A UBL document schema in one UBL namespace that is dependant upon type definitions or element declarations defined in another namespace MUST NOT import internal schema modules from that namespace.
Modularity In Action

urn:oasis:names:specification:ubl:schema:Order-1.0

Legend
- Document Schema
- Internal Schema Module
- External Schema Module
- Import
- Include
- x: y: z: urn
# Schema Modularity

<table>
<thead>
<tr>
<th>[SSM1]</th>
<th>UBL Schema expressions MAY be split into multiple schema modules.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SSM2]</td>
<td>A document schema in one UBL namespace that is dependent upon type definitions or element declarations defined in another namespace MUST only import the document schema from that namespace.</td>
</tr>
<tr>
<td>[SSM3]</td>
<td>A UBL document schema in one UBL namespace that is dependant upon type definitions or element declarations defined in another namespace MUST NOT import internal schema modules from that namespace.</td>
</tr>
<tr>
<td>[SSM4]</td>
<td>Imported schema modules MUST be fully conformant with UBL naming and design rules.</td>
</tr>
<tr>
<td>[SSM5]</td>
<td>UBL schema modules MUST either be treated as external schema modules or as internal schema modules of the document schema.</td>
</tr>
<tr>
<td>[SSM6]</td>
<td>All UBL internal schema modules MUST be in the same namespace as their corresponding document schema.</td>
</tr>
<tr>
<td>[SSM7]</td>
<td>Each UBL internal schema module MUST be named {ParentSchemaModuleName} {InternalSchemaModuleName} {schema module}</td>
</tr>
<tr>
<td>[SSM8]</td>
<td>A UBL schema module MAY be created for reusable components.</td>
</tr>
<tr>
<td>[SSM9]</td>
<td>A schema module defining all ubl:CommonAggregateComponents MUST be created.</td>
</tr>
</tbody>
</table>
### Schema Modularity

<table>
<thead>
<tr>
<th>SSM10</th>
<th>The <code>ubl:CommonAggregateComponents</code> schema module MUST be named &quot;ubl:CommonAggregateComponents Schema Module&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM11</td>
<td>A schema module defining all <code>ubl:CommonBasicComponents</code> MUST be created.</td>
</tr>
<tr>
<td>SSM12</td>
<td>The <code>ubl:CommonBasicComponents</code> schema module MUST be named &quot;ubl:CommonBasicComponents Schema Module&quot;</td>
</tr>
<tr>
<td>SSM13</td>
<td>A schema module defining all <code>ccts:CoreComponentTypes</code> MUST be created.</td>
</tr>
<tr>
<td>SSM14</td>
<td>The <code>ccts:CoreComponentType</code> schema module MUST be named &quot;ccts:CoreComponentType Schema Module&quot;</td>
</tr>
<tr>
<td>SSM15</td>
<td>The <code>xsd:facet</code> feature MUST not be used in the <code>ccts:CoreComponentType</code> schema module.</td>
</tr>
<tr>
<td>SSM16</td>
<td>A schema module defining all <code>ccts:UnspecialisedDatatypes</code> MUST be created.</td>
</tr>
<tr>
<td>SSM17</td>
<td>The <code>ccts:UnspecialisedDatatype</code> schema module MUST be named &quot;ccts:UnspecialisedDatatype Schema Module&quot;</td>
</tr>
</tbody>
</table>
General Naming

• **Top-level element:**
  – An element that encloses a whole UBL business message. Note that UBL business messages might be carried by messaging transport protocols that themselves have higher-level XML structure. Thus, a UBL top-level element is not necessarily the root element of the XML document that carries it.

• **Lower-level element:**
  – An element that appears inside a UBL business message. Lower-level elements consist of intermediate and leaf level.

• **Intermediate element:**
  – An element not at the top level that is of a complex type, only containing other elements and attributes.

• **Leaf element:**
  – An element containing only character data (though it may also have attributes). Note that, because of the XSD mechanisms involved, a leaf element that has attributes must be declared as having a complex type, but a leaf element with no attributes may be declared with either a simple type or a complex type.

• **Common attribute:**
  – An attribute that has identical meaning on the multiple elements on which it appears. A common attribute might or might not correspond to an XSD global attribute.
## General Naming Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GNR1]</td>
<td>UBL XML element, attribute and type names MUST be in the English language using the primary English spellings provided in the Oxford English Dictionary.</td>
</tr>
<tr>
<td>[GNR2]</td>
<td>UBL XML element, attribute and type names MUST be consistently derived from CCTS conformant dictionary entry names.</td>
</tr>
<tr>
<td>[GNR3]</td>
<td>UBL XML element, attribute and type names constructed from <code>ccts.DictionaryEntryNames</code> MUST NOT include periods, spaces, other separators, or characters not allowed by W3C XML 1.0 for XML names.</td>
</tr>
<tr>
<td>[GNR4]</td>
<td>UBL XML element, attribute, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions published in Appendix B.</td>
</tr>
<tr>
<td>[GNR5]</td>
<td>Acronyms and abbreviations MUST only be added to the UBL approved acronym and abbreviation list after careful consideration for maximum understanding and reuse.</td>
</tr>
<tr>
<td>[GNR6]</td>
<td>The acronyms and abbreviations listed in Appendix B MUST always be used.</td>
</tr>
<tr>
<td>[GNR7]</td>
<td>UBL XML element, attribute and type names MUST be in singular form unless the concept itself is plural.</td>
</tr>
<tr>
<td>[GNR8]</td>
<td>The UpperCamelCase (UCC) convention MUST be used for naming elements and types.</td>
</tr>
<tr>
<td>[GNR9]</td>
<td>The lowerCamelCase (LCC) convention MUST be used for naming attributes.</td>
</tr>
</tbody>
</table>
### ABIE Element and complexType Naming -

<table>
<thead>
<tr>
<th>CTN1</th>
<th>A UBL xsd:complexType name based on an ccts:AggregateBusinessInformationEntity MUST be the ccts:DictionaryEntryName with the separators removed and with the &quot;Details&quot; suffix replaced with &quot;Type&quot;.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELN1</td>
<td>A UBL global element name based on a ccts:ABIE MUST be the same as the name of the corresponding xsd:complexType to which it is bound, with the word &quot;Type&quot; removed.</td>
</tr>
</tbody>
</table>

---

#### XML Schema Model

<table>
<thead>
<tr>
<th>XML Tag Name</th>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/BBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventoryAvailabilityRequest</td>
<td>InventoryAvailabilityRequestType</td>
<td>InventoryAvailability_Request. Details</td>
<td>ABIE</td>
</tr>
</tbody>
</table>
## BBIE Property Element and complexType Naming

<table>
<thead>
<tr>
<th>ELN2</th>
<th>A UBL global element name based on an unqualified ccts:BBIEProperty MUST be the same as the name of the corresponding xsd:complexType to which it is bound, with the word &quot;Type&quot; removed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTN2</td>
<td>A UBL xsd:complexType name based on a ccts:BasicBusinessInformationEntityProperty MUST be the ccts:DictionaryEntryName shared property term and its qualifiers and the representation term of the shared ccts:BasicBusinessInformationEntity, with the separators removed and with the &quot;Type&quot; suffix appended after the representation term.</td>
</tr>
</tbody>
</table>

### XML Schema Model

<table>
<thead>
<tr>
<th>XML Tag Name</th>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/BBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>NameType</td>
<td>Supplier_ Organization. Name. Text</td>
<td>BBIE</td>
</tr>
</tbody>
</table>
BBIE Property Element and complexType Naming

[ELN2] A UBL global element name based on an unqualified ccts:BBIEProperty MUST be the same as the name of the corresponding xsd:complexType to which it is bound, with the word "Type" removed.

[ELN4] A UBL global element name based on a qualified ccts:BBIEProperty MUST be the same as the name of the corresponding xsd:complexType to which it is bound, with the qualifier prefixed and with the word "Type" removed.

XML Schema Model

<table>
<thead>
<tr>
<th>XML Tag Name</th>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/BBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>NameType</td>
<td>Supplier_ Organization. Name. Text</td>
<td>BBIE</td>
</tr>
<tr>
<td>DepartmentName</td>
<td>DepartmentNameType</td>
<td>Supplier_ Organization. Department_ Name. Text</td>
<td>BBIE</td>
</tr>
<tr>
<td>DepartmentID</td>
<td>DepartmentIDType</td>
<td>Supplier_ Organization. Department_ Identification. Identifier</td>
<td>BBIE</td>
</tr>
</tbody>
</table>
### ASBIE Element Naming

<table>
<thead>
<tr>
<th>XML Tag Name</th>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/BBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OfficialContactMailingAddress</td>
<td>N/A</td>
<td>Supplier_Organization. Official_Contact. Mailing_Address</td>
<td>ASBIE</td>
</tr>
</tbody>
</table>

**Note:** No CT for ASBIE
ASBIE Element Naming

| [ELN3] | A UBL global element name based on a qualified ccts:ASBIE MUST be the ccts:ASBIE dictionary entry name property term and its qualifiers; and the object class term and qualifiers of its associated ccts:ABIE. All ccts:DictionaryEntryName separators MUST be removed. Redundant words in the ccts:ASBIE property term or its qualifiers and the associated ccts:ABIE object class term or its qualifiers MUST be dropped. |

```
<xsd:complexType name="ForecastUpdateType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:CategoryCode>ABIE</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Forecast_Update. Details</ccts:DictionaryEntryName>
      <ccts:Definition>ForecastUpdate</ccts:Definition>
      <ccts:ObjectClass>Update</ccts:ObjectClass>
      <ccts:RepresentationTerm>TBD</ccts:RepresentationTerm>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element ref="cac:ForecastSupplierOrganization"/>
  </xsd:sequence>
</xsd:complexType>
```

Note: Truncation rules apply to ASBIEs
Unspecialized Datatypes ComplexType Naming

[CTN3] A UBL xsd:complexType for a cct:UnspecializedDatatype used in the UBL model MUST have the name of the corresponding ccts:CoreComponentType, with the separators removed and with the “Type” suffix appended.

Example:

<!-- ===== Primary Representation Term: AmountType ===== -->
<xsd:complexType name="AmountType">
  ...
</xsd:complexType>
[CTN4] A UBL xsd:complexType for a cct:UnspecializedDatatype based on a ccts:SecondaryRepresentationTerm used in the UBL model MUST have the name of the corresponding ccts:SecondaryRepresentation Term, with the separators removed and with the “Type” suffix appended.

Example:

<!-- ===== Secondary Representation Term: GraphicType ===== -->
<xsd:complexType name="GraphicType">
  ...
</xsd:complexType>
CCT ComplexType Naming

[CTN5] A UBL xsd:complexType name based on a ccts:CoreComponentType MUST be the Dictionary entry name of the ccts:CoreComponentType, with the separators removed.

Example:

<!-- ===== CCT: QuantityType ===== -->
<xsd:complexType name="QuantityType">
  ...
</xsd:complexType>
Attribute Naming

[ATN1] Each CCT:SupplementaryComponent xsd:attribute "name" MUST be the Dictionary Entry Name object class, property term and representation term of the ccts:SupplementaryComponent with the separators removed.

Example:

<table>
<thead>
<tr>
<th>ccts:SupplementaryComponent</th>
<th>ubl:attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount Currency. Identifier</td>
<td>amountCurrencyID</td>
</tr>
<tr>
<td>Measure Unit. Code</td>
<td>measureUnitCode</td>
</tr>
</tbody>
</table>
Type Definitions

[GTD1] All types MUST be named

Example:

```xml
<xsd:complexType name="QuantityType">
  ...
</xsd:complexType>
```

[GTD2] The xsd:anyType MUST NOT be used
CCT SimpleType Definition

[STD1] For every ccts:CCT whose supplementary components map directly onto the properties of a built-in xsd:Datatype, the ccts:CCT MUST be defined as a named xsd:simpleType in the ccts:CCT schema module.

Example:

<!-- ===== CCT: DateTimeType ===== -->
<xsd:simpleType name="DateTimeType">
  ...
  <xsd:restriction base="cct:DateTimeType"/>
</xsd:simpleType>
[CTD1] For every class identified in the UBL model, a named xsd:complexType MUST be named

<table>
<thead>
<tr>
<th>XML Tag Name</th>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/BBIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventoryAvailabilityRequest</td>
<td>InventoryAvailabilityRequestType</td>
<td>InventoryAvailability_ Request, Details</td>
<td>ABIE</td>
</tr>
</tbody>
</table>
[CTD2] Every ccts:ABIE xsd:complexType definition content model MUST use the xsd:sequence element with appropriate global element references, or local element declarations in the case of ID and Code, to reflect each property of its class as defined in the corresponding UBL model.

```xml
<xsd:complexType name="InventoryItemType">
  <xsd:sequence>
    <xsd:element name="InventoryItemID" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:Quantity" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:Name" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```
Complex Type Definition – BBIE Property

[CTD3] Every ccts:BBIEProperty xsd:complexType definition content model MUST use the xsd:simpleContent element.

```xml
<xsd:complexType name="DepartmentNameType">
  <xsd:simpleContent>
    <xsd:extension base="udt:TextType"/>
  </xsd:simpleContent>
</xsd:complexType>
```


[CTD5] Every ccts:BBIEProperty xsd:complexType content model xsd:base attribute value MUST be the ccts:CCT of the unspecialized or specialized UBL Datatype as appropriate.
Core Component type – Type Definitions

For every ccts:CCT whose supplementary components are not equivalent to the properties of a built-in xsd:Datatype, the ccts:CCT MUST be defined as a named xsd:complexType in the ccts:CCT schema module.

Example:

```xml
<xsd:complexType name="QuantityType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:decimal">
            <xsd:attribute name="quantityUnitCode" type="xsd:normalizedString" use="optional"/>
            <xsd:attribute name="quantityUnitCodeListID" type="xsd:normalizedString" use="optional"/>
            <xsd:attribute name="quantityUnitCodeListAgencyID" type="xsd:normalizedString" use="optional"/>
            <xsd:attribute name="quantityUnitCodeListAgencyName" type="xsd:string" use="optional"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```
Handling Supplementary Components

[CTD16] Each CCT:SupplementaryComponent xsd:attribute “type” MUST define the specific xsd:Built-inDatatype or the user defined xsd:simpleType for the ccts:SupplementaryComponent of the ccts:CCT.

Example:

```xml
<xsd:attribute name="measureUnitCode"
    type="xsd:normalizedString" use="required"/>
```
Datatype complex and simpleType Definitions

- There is a direct one-to-one relationship between ccts:CoreComponentTypes and ccts:PrimaryRepresentationTerms
  - several ccts:SecondaryRepresentationTerms that are subsets of their parent ccts:PrimaryRepresentationTerm
- The total set of ccts:RepresentationTerms by their nature represent ccts:Datatypes
- For each ccts:PrimaryRepresentationTerm or ccts:SecondaryRepresentationTerm, a ccts:UnspecializedDatatype exists
- These ccts:UnspecializedDatatypes are expressed as complex or simple types that are of the type of its corresponding ccts:CoreComponentType.

[CTD6] For every Datatype used in the UBL model, a named xsd:complexType or xsd:simpleType MUST be defined.
Datatype complexType and simpleType Definitions

[CTD7] Every unspecialized Datatype must be based on a ccts:CCT represented in the CCT schema module, and must represent an approved primary or secondary representation term identified in the CCTS.

[CTD8] Each unspecialized Datatype xsd:complexType must be based on its corresponding CCT xsd:complexType.

[CTD9] Every unspecialized Datatype that represents a primary representation term whose corresponding ccts:CCT is defined as an xsd:simpleType MUST also be defined as an xsd:simpleType and MUST be based on the same xsd:simpleType.

[CTD10] Every unspecialized Datatype that represents a secondary representation term whose corresponding ccts:CCT is defined as an xsd:simpleType MUST also be defined as an xsd:simpleType and MUST be based on the same xsd:simpleType.
[ELD3] For every class identified in the UBL model, a global element bound to the corresponding xsd:complexType MUST be declared.

Example:

For the BuyerParty. Details object class, a complex type/global element declaration pair is created through the declaration of a Party element that is of type BuyerPartyType.

```
<xsd:element name="BuyerParty" type="BuyerPartyType"/>
<xsd:complexType name="BuyerPartyType">
    ...
</xsd:complexType>
```
When a `ccts:ASBIE` is **unqualified**, it is bound via reference to the global `ccts:ABIE` element to which it is associated. When an `ccts:ASBIE` is qualified, a new element MUST be declared and bound to the `xsd:complexType` of its associated `ccts:AggregateBusinessInformationEntity`.

```xml
<xsd:complexType name="InventoryAvailabilityRequestType">
    <xsd:annotation>
        <xsd:documentation>
            <ccts:CategoryCode>ABIE</ccts:CategoryCode>
            <ccts:DictionaryEntryName/>
            <ccts:Definition>This holds information specific for an inventory availability request.</ccts:Definition>
            <ccts:ObjectClass/>
            <ccts:RepresentationTerm/>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element ref="cac:InventoryOrganization"/>
        <xsd:element ref="cac:InventoryItem" maxOccurs="unbounded"/>
        <xsd:element ref="cac:SupplierOrganization"/>
    </xsd:sequence>
</xsd:complexType>
```
When an `xsd:complexType` is unqualified, it is bound via reference to the global `xsd:complexType` of the associated `xsd:complexType`. When an `xsd:complexType` is **qualified**, a new element MUST be declared and bound to the `xsd:complexType` of its associated `xsd:complexType`.

```xml
<xsd:complexType name="ForecastUpdateType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:CategoryCode>ABIE</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Forecast_Update.</ccts:DictionaryEntryName>
      <ccts:Definition>ForecastUpdate</ccts:Definition>
      <ccts:ObjectClass>Update</ccts:ObjectClass>
      <ccts:RepresentationTerm>TBD</ccts:RepresentationTerm>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element ref="cac:ForecastSupplierOrganization"/>
    <xsd:annotation> ......<xsd:annotation>
  </xsd:sequence>
</xsd:complexType>
```

**ASBIE**
Global elements declared for Qualified BBIE Properties must be of the same type as its corresponding Unqualified BBIE Property. (i.e. Property Term + Representation Term.)

Example:

```xml
<xsd:element name="AdditionalStreetName"
    type="cbc:StreetNameType"/>
```
[ATD1] User defined attributes SHOULD NOT be used. When used, user defined attributes MUST only convey CCT:SupplementaryComponent information.
[CDL1] All UBL Codes MUST be part of a UBL or externally maintained Code List.

[CDL2] The UBL Library SHOULD identify and use external standardized code lists rather than develop its own UBL-native code lists.

[CDL3] The UBL Library MAY design and use an internal code list where an existing external code list needs to be extended, or where no suitable external code list exists.
[GXS2] UBL MUST provide two normative schemas for each transaction. One schema shall be fully annotated. One schema shall be a run-time schema devoid of documentation.
Documentation Rules

• Every CCTS construct must contain all mandatory CCTS Section 7 storage metadata

• Every element declaration and type definition in a UBL model must include all mandatory CCTS Section 7 Storage metadata

• Example:

[DOC1] Every data type definition MUST contain a structured set of annotations in the following sequence and pattern:

  – UniqueIdentifier (mandatory): The identifier that references a data type instance in a unique and unambiguous way.
  – CategoryCode (mandatory): The category to which the object belongs. For example, BBIE, ABIE, ASBIE.
  – DictionaryEntryName (mandatory): The official name of a data type.
  – Definition (mandatory): The semantic meaning of a data type.
  – Version (mandatory): An indication of the evolution over time of a data type instance.
  – QualifierObjectClass (optional): The qualifier for the object class.
  – Usage Rule (optional, repetitive): A constraint that describes specific conditions that are applicable to the data type.
Implementation Models

Business Operations view
UML and spreadsheets
Design

Analysis
XML Schema Libraries
EDI Message Definitions
Database Schemas
Business Document Models

Functional
Final presentation is application dependent (e.g. Stylesheet)

The Real World
Messages/Documents

Limited interoperability
Greater interoperability
Formatting Specifications

• Formatting Specification in Detail
  – A formatting specification is a recipe for a stylesheet, but is not in and of itself a transformation script.
  – Writers of stylesheets, programs, or any other open and proprietary transformation technologies rely on formatting specifications for direction regarding content identification and layout.

• PDF Renderings of example instances
Example Formatting Specification

- Despatch advice formatting specifications
  - Three sample formatting specifications are offered for this document type:
    - Office-oriented despatch advice form
    - Joinery-oriented despatch advice form
    - United Nations Layout Key form 351: Despatch Advice
Sample UBL Document Instance

**Invoice**
Invoice Number: 9834562  
Invoice Date: 02-14-03  
Purchase Order No: 20031234-1  
Sales Order Number: 154135798  
Shipment Date: 02-14-03

**To:** Bills Microdevices  
413 Spring St.  
Elgin, Ill 60123

**From:** Joes Office Supply  
32 W. Lakeshore Dr  
Chicago, Ill 60022

**Billing**  
**Contact:** Melanie Farber (312) 865-2199

Shipped to: 413 N Spring St.  
Elgin, Ill 60123

<table>
<thead>
<tr>
<th>Line Num</th>
<th>Part Number</th>
<th>Description</th>
<th>Qty</th>
<th>Unit Price</th>
<th>Extended Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32145-12</td>
<td>Pencils, box #2 red</td>
<td>5</td>
<td>$2.50</td>
<td>$12.50</td>
</tr>
<tr>
<td>2</td>
<td>78-697-24</td>
<td>Xerox Paper- case</td>
<td>12</td>
<td>$30.00</td>
<td>$360.00</td>
</tr>
<tr>
<td>3</td>
<td>091356-3</td>
<td>Pens, box, blue finepoint</td>
<td>10</td>
<td>$5.00</td>
<td>$50.00</td>
</tr>
<tr>
<td>4</td>
<td>543-165-1</td>
<td>Tape, tin case</td>
<td>3</td>
<td>$12.50</td>
<td>$37.50</td>
</tr>
<tr>
<td>5</td>
<td>984567-12</td>
<td>Staples, wire, box</td>
<td>10</td>
<td>$1.00</td>
<td>$10.00</td>
</tr>
<tr>
<td>6</td>
<td>091344-5</td>
<td>Pens, box red felt tip</td>
<td>5</td>
<td>$4.50</td>
<td>$22.50</td>
</tr>
<tr>
<td>7</td>
<td>21457-3</td>
<td>Mousepad, blue</td>
<td>10</td>
<td>$0.50</td>
<td>$5.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tax</strong></td>
<td></td>
<td></td>
<td><strong>$47.95</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total Due</strong></td>
<td></td>
<td></td>
<td><strong>$527.45</strong></td>
</tr>
</tbody>
</table>

UBL Document Component Model
UBL Component Packages

- Address Contract
- Delivery
- Document Reference
- Hazardous Item
- Item
- Party
- Payment
- Procurement
- Tax
Procurement Package

## Order Document Spreadsheet Snippet

<table>
<thead>
<tr>
<th>UBL Name</th>
<th>Dictionary Entry Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Order. Details</td>
</tr>
<tr>
<td>BuyersID</td>
<td>Order. Buyers Identifier. Identifier</td>
</tr>
<tr>
<td>SellersID</td>
<td>Order. Sellers Identifier. Identifier</td>
</tr>
<tr>
<td>CopyIndicator</td>
<td>Order. Copy. Indicator</td>
</tr>
<tr>
<td>GUID</td>
<td>Order. Globally Unique Identifier</td>
</tr>
<tr>
<td>IssueDate</td>
<td>Order. Issue Date. Date</td>
</tr>
<tr>
<td>Note</td>
<td>Order. Note. Text</td>
</tr>
<tr>
<td>EarliestDate</td>
<td>Order. Earliest Date. Date</td>
</tr>
<tr>
<td>ExpiryDate</td>
<td>Order. Expiry Date. Date</td>
</tr>
<tr>
<td>ValidityDurationMeasure</td>
<td>Order. Validity Duration. Measure</td>
</tr>
<tr>
<td>TaxTotalAmount</td>
<td>Order. Tax Total. Amount</td>
</tr>
<tr>
<td>LineExtensionTotalAmount</td>
<td>Order. Line. Extension Total. Amount</td>
</tr>
<tr>
<td>TotalPackagesQuantity</td>
<td>Order. Total. Packages Quantity. Quantity</td>
</tr>
<tr>
<td>GrossWeightMeasure</td>
<td>Order. Gross. Weight. Measure</td>
</tr>
<tr>
<td>NetWeightMeasure</td>
<td>Order. Net. Weight. Measure</td>
</tr>
<tr>
<td>NetNetWeightMeasure</td>
<td>Order. Net Net. Weight. Measure</td>
</tr>
</tbody>
</table>
Implementation Model

• XSD Schema
  - Order
  - Order Response
  - Order Response Simple
  - Order Change
  - Order Cancellation
  - Despatch Advice
  - Receipt Advice
  - Invoice

• Code List Schema
  - Acknowledgement Response Code
  - Allowance Charge Reason Code
  - Channel Code
  - Chip Code
  - Country Identification Code
  - Currency Code
  - Document Status Code
  - Latitude Direction Code
  - Line Status Code
  - Longitude Direction Code
  - Operator Code
  - Payment Means Code
  - Substitution Status Code
Implementation Model

- Reusable Component Class Diagrams
  - Autogenerated from XSD Schema using Ontogenics hyperModel*
    - Address
    - Contract
    - Despatch Line
    - Document Reference
    - ‘Hazardous Item
    - Item
    - Party
    - Payment
    - Procurement
    - Shipment
    - Tax

*View David Carlson’s work at: http://www.xmlmodeling.com/
Creating UBL Schema
BIEs as Schema Content

- ABIE contains/aggregates ASBIEs and/or BBIEs

There’s a bit more to it than this.
ABIEs as XML Schema

- ABIE → complexType
- ABIE aggregates elements (ASBIEs and BBIEs)

```xml
<xsd:complexType name="TBD">
  <xsd:sequence>
    <xsd:element ref="TBD"/>
    <xsd:element ref="TBD"/>
    <xsd:element ref="TBD"/>
    <xsd:element ref="TBD"/>
  </xsd:sequence>
</xsd:complexType>
```

- BBIEs are INtrinsic to ABIE
- ASBIEs are INtrinsic to ABIE

See UBL Rule(s):
- CTN1
- ELN1
ABIEs as XML Schema: the UBL Rule(s)

- 2 Schema representations for an ABIE
  - One is a complexType
  - The other is an element

[CTN1] A UBL:xsd:complexType name based on a ccts:AggregateBusinessInformationEntity MUST be the ccts:DictionaryEntryName with the separators removed and with the “Details” suffix replaced with “Type”.

[ELN1] A UBL:global element name based on a ccts:ABIE MUST be the same as the name of the corresponding xsd:complexType to which it is bound, with the word “Type” removed.
BBIEs as XML Schema

- BBIE → ref within an ABIE
- BBIE Property → element declaration

This distinction is important!

BBIEs are **IN**trinsic to ABIE
- BBIE Properties are **EX**trinsic to ABIE
- BBIE Properties are linked to either UDT or QDT
- So, is a BBIE and a BBIE Property the same thing? No.

See UBL Rule(s):
- CTN2
- ELN2
BBIEs as XML Schema: the UBL Rule(s)

- 2 Schema representations for a BBIE
  - One is a complexType
  - The other is an element

[CTN2] A UBL xsd:complexType name based on a
ccts:BasicBusinessInformationEntityProperty MUST be the
ccts:DictionaryEntryName shared property term and its qualifiers and
representation term of the shared ccts:BasicBusinessInformation-
Entity, with the separators removed and with the “Type” suffix appended
after the representation term.

[ELN2] A UBL global element name based on an unqualified ccts:BBIEProperty
MUST be the same as the name of the corresponding xsd:complexType to
which it is bound, with the word “Type” removed.
ASBIEs as XML Schema

- Similar in structure to BBIEs
- ASBIE’s → element refs

```xml
<xsd:complexType name="TBD">
  <xsd:sequence>
    <xsd:element ref="TBD"/>
    <xsd:element ref="TBD"/>
    <xsd:element ref="TBD"/>
  </xsd:sequence>
</xsd:complexType>
```

- Each ASBIE/element ref(ernce)s an EXtrinsic element that is of an ABIE complexType.

Critical!
Each ASBIE/element refers to an extrinsic element that is ‘of’ an ABIE complexType.
ASBIEs as XML Schema: the UBL Rule(s)

is an association between two classes.

As such, an element representing the `ccts:AssociationBusinessInformationEntity` does not have its own unique `xsd:ComplexType`. Instead, when an element representing a `ccts:AssociationBusinessInformationEntity` is declared, the element is bound to the `xsd:complexType` of its associated `ccts:AggregateBusinessInformationEntity`.
ASBIEs as XML Schema: the UBL Rule(s)

1 Schema representation for an ASBIE

[ELN3] A UBL global element name based on a qualified `ccts:ASBIE` MUST be the `ccts:ASBIE` dictionary entry name property term and its qualifiers; and the object class term and qualifiers of its associated `ccts:ABIE`. All `ccts:DictionaryEntryName` separators MUST be removed. Redundant words in the `ccts:ASBIE` property term or its qualifiers and the associated `ccts:ABIE` object class term or its qualifiers MUST be dropped.
Supporting Data Types

• Qualified Data Types (QDTs)
  – Derived from UDTs
  – With restrictions (on the Content Component or Supplementary Component)

• Unqualified Data Types (UDTs)
  – Derived from Core Component Types
  – With NO restrictions (on the Content Component or Supplementary Component)

• Let’s look at some specific examples...
# QDT – Review of Syntax Neutral

- Based on Unqualified Data Types (UDT)
- Specialization of UDT

<table>
<thead>
<tr>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/AS BIE/BBIE</th>
<th>Object Class Qualifier</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Data Type Dictionary Entry Name</th>
<th>Core Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailing_Address_Details</td>
<td>ABIE</td>
<td>Mailing</td>
<td>Address</td>
<td>Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mailing_Address_Line_One_Text</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>LineOne</td>
<td>Text</td>
<td></td>
<td>TextType</td>
<td>Text. Type</td>
<td>Text</td>
</tr>
<tr>
<td>Mailing_Address_Line_Two_Text</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>LineTwo</td>
<td>Text</td>
<td></td>
<td>TextType</td>
<td>Text. Type</td>
<td>Text</td>
</tr>
<tr>
<td>Mailing_Address_City_Name_Text</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>CityName</td>
<td>Text</td>
<td></td>
<td>TextType</td>
<td>Text. Type</td>
<td>Text</td>
</tr>
<tr>
<td>Mailing_Address_Postcode_Code</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>Postcode</td>
<td>Code</td>
<td></td>
<td>Postal</td>
<td>CodeType</td>
<td>Postal_Code. Type</td>
</tr>
</tbody>
</table>

**UDT Code Type** is derived from **QDT**
QDT – Schema Syntax Specific

- BBIE Properties are linked to either UDT or QDT
- When linked to QDT... it’s b/c of some restriction(s)

```xml
<xsd:complexType name="PostalCodeType">
    <xsd:annotation>
        <xsd:documentation>
            <ccts:ComponentType>TBD</ccts:ComponentType>
            <ccts:DictionaryEntryName>TBD</ccts:DictionaryEntryName>
            <ccts:Definition>TBD</ccts:Definition>
            <ccts:ObjectClass>TBD</ccts:ObjectClass>
            <ccts:RepresentationTerm>TBD</ccts:RepresentationTerm>
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:restriction base="udt:CodeType">
            <xsd:minLength value="1"/>
            <xsd:maxLength value="5"/>
        </xsd:restriction>
    </xsd:simpleContent>
</xsd:complexType>
```
UDT – Review of Syntax Neutral

- ‘Predefined’ Schema Module
- Based Directly on Core Component Types (CCTs)
- Used as the basis for Qualified Data Types

<table>
<thead>
<tr>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE AS BE/BBIE</th>
<th>Object Class Qualifier</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Unqualified Data Type</th>
<th>Dictionary Entry Name</th>
<th>Data Type Dictionary Entry Name</th>
<th>Core Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailing Address, Details</td>
<td>ABIE</td>
<td>Mailing</td>
<td>Address</td>
<td>Details</td>
<td>Details</td>
<td>Details</td>
<td>Details</td>
<td>TextType</td>
<td>TextType</td>
<td>Text</td>
<td>TextType</td>
</tr>
<tr>
<td>Mailing Address, Line One, Text</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>LineOne</td>
<td>Text</td>
<td>TextType</td>
<td>TextType</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mailing Address, Line Two, Text</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>LineTwo</td>
<td>Text</td>
<td>TextType</td>
<td>TextType</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mailing Address, City Name, Text</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>CityName</td>
<td>Text</td>
<td>TextType</td>
<td>TextType</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mailing Address, Postcode, Code</td>
<td>BBIE</td>
<td>Mailing</td>
<td>Address</td>
<td>Postcode</td>
<td>Code</td>
<td>Postalcde</td>
<td>CodeType</td>
<td>Postcode, Code</td>
<td>CodeType</td>
<td>Postcode, Code</td>
<td>CodeType</td>
</tr>
</tbody>
</table>

Rep Term & Normalized CCT is derived from UDT Text Text. Type
UDT – Schema Syntax Specific

• BBIE Properties are linked to either UDT or QDT
• When linked to UDT...
  – restriction(s) may NOT exist
  – extensions should be used

```xml
<xsd:complexType name="DepartmentNameType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:ComponentType></ccts:ComponentType>
      <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
      <ccts:Definition></ccts:Definition>
      <ccts:ObjectClass></ccts:ObjectClass>
      <ccts:RepresentationTerm></ccts:RepresentationTerm>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="udt:TextType"/>
  </xsd:simpleContent>
</xsd:complexType>
```

See UBL Rule(s):
CTD3
CTD4
CTD5
5.1.3.2 Basic Business Information Entities

Basic Business Information Entities (BBIEs), in accordance with the Core Components Technical Specification, always have a primary representation term, and may have secondary representation terms, which describes their structural representation. These representation terms are expressed in the UBL Model as Uns specialised Datatypes bound to a Core Component Type that describes their structure. In addition to the unspecialised Datatypes defined in CCTS, UBL has defined a set of specialised Datatypes that are derived from the CCTS unqualified Datatypes. There are a set of rules concerning the way these relationships are expressed in the UBL XML library. As discussed above, BBIE properties are represented with complex types. Within these are simpleContent elements that extend the Datatypes.

[CTD3] Every ccts:BBIEProperty xsd:complexType definition content model MUST use the xsd:simpleContent element.


[CTD5] Every ccts:BBIEProperty xsd:complexType content model xsd:base attribute value MUST be the ccts:CCT of the unspecialised or specialised UBL Datatype as appropriate.
So, what does a UDT look like?

- UDTs are based on Schema ‘built-in’ data types

See UBL Rule(s):
CTD7 - CTD12
5.1.3.3.1 Uns specialised Datatypes

The `ccts:UnspecialisedDatatypes` reflect the instantiation of the `ccts:CoreComponentTypes`. Each `ccts:UnspecialisedDatatype` declaration is based on its corresponding qualified `ccts:CoreComponentType` and represents either a primary or secondary representation term.

<table>
<thead>
<tr>
<th>CTD7</th>
<th>Every unspecialised Datatype must be based on a <code>ccts:CCT</code> represented in the CCT schema module, and must represent an approved primary or secondary representation term identified in the CCTS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTD8</td>
<td>Each unspecialised Datatype <code>xsd:complexType</code> must be based on its corresponding CCT <code>xsd:complexType</code>.</td>
</tr>
<tr>
<td>CTD9</td>
<td>Every unspecialised Datatype that represents a primary representation term whose corresponding <code>ccts:CCT</code> is defined as an <code>xsd:simpleType</code> MUST also be defined as an <code>xsd:simpleType</code> and MUST be based on the same <code>xsd:simpleType</code>.</td>
</tr>
<tr>
<td>CTD10</td>
<td>Every unspecialised Datatype that represents a secondary representation term whose corresponding <code>ccts:CCT</code> is defined as an <code>xsd:simpleType</code> MUST also be defined as an <code>xsd:simpleType</code> and MUST be based on the same <code>xsd:simpleType</code>.</td>
</tr>
<tr>
<td>CTD11</td>
<td>Each unspecialised Datatype <code>xsd:complexType</code> definition must contain one <code>xsd:simpleContent</code> element.</td>
</tr>
<tr>
<td>CTD12</td>
<td>The unspecialised Primary Representation Term Datatype <code>xsd:complexType</code> definition <code>xsd:simpleContent</code> element must contain one <code>xsd:restriction</code> element with an <code>xsd:base</code> attribute whose value is equal to the corresponding <code>ccts:complexType</code>.</td>
</tr>
</tbody>
</table>
BIEs – A Specific Example

- Now.... let’s look at some specific ABIE/BBIE/ASBIE examples!
- We’ll be putting all of this to use in the ‘Step By Step’ session
- Between now and then we’ll build on these ABIE/BBIE/ASBIE concepts by outlining their places in the Schema hierarchy
  - in the ‘Schema Modularity’ session
Example Source

- Locate the ‘Supplier Organization’ ABIE in your course spreadsheet
  - value for (column ‘H’) is: ‘Supplier_ Organization. Details’

<table>
<thead>
<tr>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/BBIE</th>
<th>Object Class Term</th>
<th>Property Term</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Unqualified Data Type</th>
<th>Data Type Dictionary Entry Name</th>
<th>Core Component Type (CCT)</th>
<th>Associate Object Class Qualifier</th>
<th>Associated Object Class Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupplierOrganizationType</td>
<td>Supplier_Organization. Details</td>
<td>ABIE</td>
<td>Organization</td>
<td>Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NameType</td>
<td>Supplier_Organization. Name. Text</td>
<td>BBIE</td>
<td>Organization</td>
<td>Name</td>
<td>Text</td>
<td>TextType</td>
<td>Text. Type</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DepartmentNameType</td>
<td>Supplier_Organization. Department_Name. Text</td>
<td>BBIE</td>
<td>Organization</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DepartmentIDType</td>
<td>Supplier_Organization. Department_Identifier. Identifier</td>
<td>BBIE</td>
<td>Organization</td>
<td>Department</td>
<td>Identification</td>
<td>Identifier</td>
<td>Department_Identifier. Type</td>
<td>Identifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier_Organization. Official_Contact. Mailing_Address</td>
<td>ASBIE</td>
<td>Organization</td>
<td>Mailing</td>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier_Organization. Availability. Inventory_Item</td>
<td>ASBIE</td>
<td>Organization</td>
<td>Availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**ABIE: Supplier Organization. Details**

- **ABIE → complexType**

<table>
<thead>
<tr>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBI EBBIE</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Unqualified Data Type</th>
<th>Data Dictionary Entry Name</th>
<th>Core Component Type (CCT)</th>
<th>Associated Object Class Qualifier</th>
<th>Associated Object Class Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupplierOrganizationType</td>
<td>Supplier Organization, Details</td>
<td>ABIE</td>
<td>Organization</td>
<td>Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
<xsd:complexType name="SupplierOrganizationType">
  <xsd:annotation> ...
  <xsd:sequence>
    <xsd:element ref="cbc:Name" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:DepartmentName" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:DepartmentID" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cac:MailingAddress" maxOccurs="unbounded"/>
    <xsd:element ref="cac:InventoryItem" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

©Mark Crawford, LMI Government Consulting, April 2005
ABIE: Supplier Organization. Details

- Now, let’s look at the 1st of these 3 BBIEs
- And its associated BBIE Property, of course

```xml
<xsd:complexType name="SupplierOrganizationType">
  <xsd:annotation/>
  <xsd:sequence>
    <xsd:element ref="cbc:Name" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:DepartmentName" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cbc:DepartmentID" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="cac:MailingAddress" maxOccurs="unbounded"/>
    <xsd:element ref="cac:InventoryItem" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```
**BBIE: Supplier Organization. Name. Text**

<table>
<thead>
<tr>
<th>XML ComplexType</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBI E/BBIE</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Unqualified Data Type</th>
<th>Data Type Dictionary Entry Name</th>
<th>Core Component Type (CCT)</th>
<th>Associate Object Class Qualifier</th>
<th>Associated Object Class Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameType</td>
<td>Supplier Organization. Name. Text</td>
<td>BBIE</td>
<td>Organization</td>
<td>Name</td>
<td>Text</td>
<td></td>
<td>TextType</td>
<td>TextType</td>
<td>TextType</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
<xsd:element name="Name" type="cbc:NameType">
  <xsd:complexType name="NameType">
    <xsd:simpleContent>
      <xsd:extension base="udt:TextType"/>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

**here’s the UDT from earlier**

**BBIE Property**

©Mark Crawford, LMI Government Consulting, April 2005
• Now let’s back up to the ABIE again
• And look at the 1st of the 2 ASBIEs

```xml
<xs:complexType name="SupplierOrganizationType">
  <xs:annotation/>
  <xs:sequence>
    <xs:element ref="cbc:Name" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="cbc:DepartmentName" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="cbc:DepartmentID" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="cac:MailingAddress" maxOccurs="unbounded"/>
    <xs:element ref="cac:InventoryItem" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```
## ASBIE: Supplier Organization. Official Contact. Mailing Address

<table>
<thead>
<tr>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBIE/EBBIE</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Unqualified Data Type</th>
<th>Data Type Dictionary Entry Name</th>
<th>Core Component Type (CCT)</th>
<th>Associate Object Class Qualifier</th>
<th>Associated Object ClassTerm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
<xsd:element name="MailingAddress" type="cac:MailingAddressType">
<xsd:complexType name="MailingAddressType">
<xsd:annotation>...
<xsd:sequence>
<xsd:element ref="cbc:LineOne" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element ref="cbc:LineTwo" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element ref="cbc:CityName" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element ref="cbc:PostalCode" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```
Message Assembly

- Message Assemblies represent the base entities (i.e., root elements) that are passed in an XML message

SI-Enterprise-InventoryAvailabilityRequest

SI-Enterprise-InventoryAvailabilityResponse

A few things live here

All else lives here

references
Schema Documentation – At a Glance

- Schema Documentation
  - Not to be confused with ‘User Guides’
  - Then what is it?

The UBL NDR Provides a comprehensive set of Schema Documentation Rules that instantiate the storage rules from Section 7 of ISO 1500-5

- Let’s look at this in a Schema file?...
Schema Documentation

- Much of the XSD content in these course slides has omitted annotation / documentation (for brevity)
- Are ‘annotations’ and ‘documentation’ the same thing? No.

```xml
<xsd:complexType name="SupplierOrganizationType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:ComponentType>ABIE</ccts:ComponentType>
      <ccts:DictionaryEntryName>Supplier Organization. Details</ccts:DictionaryEntryName>
      <ccts:Definition>This holds all pertinent information relating to a supplier organization.</ccts:Definition>
      <ccts:ObjectClass>Organization</ccts:ObjectClass>
    </xsd:documentation>
  </xsd:annotation>
</xsd:complexType>
```

- Let’s look at this mapped back to the spreadsheet...
### XSD Documentation

<table>
<thead>
<tr>
<th>XML ComplexType Name</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ASBI E/BBIE</th>
<th>Object Class Term</th>
<th>Property Term Qualifier</th>
<th>Property Term</th>
<th>Representation Term</th>
<th>Data Type Qualifier</th>
<th>Unqualified Data Type</th>
<th>Data Type Dictionary Entry Name</th>
<th>Core Component Type (CCT)</th>
<th>Associate Object Class Qualifier</th>
<th>Associated Object Class Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupplierOrganizationType</td>
<td>Supplier_ Organization. Details</td>
<td>ABIE</td>
<td>Organization</td>
<td>Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
<xsd:complexType name="SupplierOrganizationType">
    <xsd:annotation>
        <xsd:documentation>
            <ccts:ComponentType>ABIE</ccts:ComponentType>
            <ccts:DictionaryEntryName>Supplier_ Organization. Details</ccts:DictionaryEntryName>
            <ccts:Definition>This holds all pertinent information relating to a supplier organization.</ccts:Definition>
            <ccts:ObjectClass>Organization</ccts:ObjectClass>
            <ccts:RepresentationTerm></ccts:RepresentationTerm>
        </xsd:documentation>
    </xsd:annotation>
    ...
</xsd:complexType>
```
The following rule describes the documentation requirements for each Aggregate Business Information Entity definition.

<table>
<thead>
<tr>
<th>[DOC5]</th>
<th>The <code>xsd:documentation</code> element for every Aggregate Business Information Entity MUST contain a structured set of annotations in the following sequence and pattern:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ComponentType (mandatory): The type of component to which the object belongs. For Aggregate Business Information Entities this must be “ABIE”.</td>
<td></td>
</tr>
<tr>
<td>• DictionaryEntryName (mandatory): The official name of the Aggregate Business Information Entity.</td>
<td></td>
</tr>
<tr>
<td>• Version (optional): An indication of the evolution over time of the Aggregate Business Information Entity.</td>
<td></td>
</tr>
<tr>
<td>• Definition (mandatory): The semantic meaning of the Aggregate Business Information Entity.</td>
<td></td>
</tr>
<tr>
<td>• ObjectClassQualifier (optional): The qualifier for the object class.</td>
<td></td>
</tr>
<tr>
<td>• ObjectClass (mandatory): The Object Class represented by the Aggregate Business Information Entity.</td>
<td></td>
</tr>
<tr>
<td>• AlternativeBusinessTerms (optional): Any synonym terms under which the Aggregate Business Information Entity is commonly known and used in the business.</td>
<td></td>
</tr>
</tbody>
</table>
Arranging the content

• Establish a consistent approach to arranging the content within ALL of your Schemas
• Will facilitate locating XSD content and can prevent errors any Schema

header info

ABIE
ABIE
ABIE
ABIE
BBIE
BBIE
BBIE
ASBIE
ASBIE
ASBIE
ASBIE

best to organize these BIEs
Referencing External Content

• By using the **xsd:include** and **xsd:import** directives
• Use these when the Schema content needed (ref’d) is located in another file / namespace

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema

targetNamespace="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmlns:cac="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
xmlns:ccts="urn:un:unece:uncefact:documentation:corecomponenttechnicalspecificati
on:2.0"

<xsd:import
namespace="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
schemaLocation="SI-Enterprise-CommonAggregateComponents.xsd"/>

...  
• This leads us into the next topic...
```
Schema Modularity

- CodeList_CurrencyC
- CodeList_Language
- CodeList_MIMEMedia
- CodeList_UnitCode

Common

UnqualifiedDataTypes-

Enterprise-QualifiedDataTypes

CommonBasicComponents

CommonAggregateComponents

Document

InventoryAvailabilityRequest

InventoryAvailabilityResponse
A word about the word ‘Document’

• A Document Schema represents the root level content definition
  – lowest or highest... depending on your proclivity

Document Schema Module

is a

SI-Enterprise-InventoryAvailabilityRequest

is a

SI-Enterprise-InventoryAvailabilityResponse

• Document does not denote / connote a ‘narrative’ document
• These XML ‘document’ Schemas define XML transactions for exchange between app servers

Important terminology
Create Document Schema header

Message Assembly

Document Schema Module

```xml
<xsd:schema targetNamespace="...">
  ABIE placeholder
  BBIE placeholder
  ASBIE placeholder
</xsd:schema>
```
Create Document Schema header

- Create the Schema root

- Define the targetNamespace
  - Define the W3C XML Schema namespace
  - Define all other namespace(s) *
  - Define the default namespace

- Set the version

- Create imports *
  - Create includes *

* when applicable
Create Document Schema header

- Create the Schema root

Let’s fill in this Schema header
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema>

</xsd:schema>
Create Document Schema header

- Create the Schema root
- Define the targetNamespace
- Define the W3C XML Schema namespace
- Define all other namespace(s) *
- Define the default namespace
<?xml version="1.0" encoding="UTF-8"?><xsd:schema

xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmlns="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmns:cac="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
xmns:ccts="urn:un:unece:unecfact:documentation:corecomponenttechnicalspecification:2.0"
Create Document Schema header

- Create the Schema root
- Define the targetNamespace
- Define the W3C XML Schema namespace
  - Define all other namespace(s) *
  - Define the default namespace
- Set the version
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

targetNamespace="urn:us:com:supplyinventory:inventorydepartment:1.0"

xmlns="urn:us:com:supplyinventory:inventorydepartment:1.0"

xmlns:cac="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"

xmlns:ccts="urn:un:unece:uncefact:documentation:corecomponenttechnicalspecificatio

n:2.0"


elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0" >
Create Document Schema header

- Create the Schema root
- Define the targetNamespace
- Define the W3C XML Schema namespace
  - Define all other namespace(s) *
  - Define the default namespace
- Set the version
  - Create imports *
  - Create includes *

<table>
<thead>
<tr>
<th>Schema root</th>
<th>Namespaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import(s) / Include(s)</td>
<td>Version</td>
</tr>
</tbody>
</table>
Document Header for: Inventory Availability Request

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema

xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmlns:cac="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
xmlns:ccts="urn:un:unece:uncefact:documentation:corecomponenttechnicalspecification:2.0"

elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0" >

<xsd:import
namespace="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
schemaLocation="SI-Enterprise-CommonAggregateComponents.xsd"/>

... 
</xsd:schema>

it’s now time to populate the Schema body...
UBL Rules for XSD headers

[GXSI1] UBL Schema MUST conform to the following physical layout as applicable:

XML Declaration
<!-- ------- Copyright Notice ------- -->
“Copyright © 2001-2004 The Organization for the Advancement of Structured Information Standards (OASIS). All rights reserved.
<!-- ------- xsd:schema Element With Namespaces Declarations ------- -->
xsd:schema element to include version attribute and namespace declarations in the following order:

  xmlns:xsd
  Target namespace
  Default namespace
  CommonAggregateComponents
  CommonBasicComponents
  CoreComponentTypes
  Unsolicited Datatypes
  Specialised Datatypes
  Identifier Schemes
  Code Lists

Attribute Declarations – elementFormDefault=”qualified”
  attributeFormDefault=”unqualified”
<!-- ------- Imports ------- -->
CommonAggregateComponents schema module
CommonBasicComponents schema module
Unspecialised Types schema module
Specialised Types schema module

Course materials follow this sequence
(as defined in the UBL NDR)
One last word about Schema modularity

The Schema headers are the 'control center' of Schema modularity

```xml
<xml version="1.0" encoding="UTF-8"?'>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmlns="urn:us:com:supplyinventory:inventorydepartment:1.0"
xmLns: cac="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
xmLns:ccts="urn:un:unece:uncefact:documentation:corecomponenttechnicalspecification:2.0"
xml: elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">

<xsd:import namespace="urn:us:com:supplyinventory:enterprise:commonaggregatecomponents"
schemaLocation="SI-Enterprise-CommonAggregateComponents.xsd"/>

... ...

</xsd:schema>
```
Schema Modules (revisited)

Schema header + Schema body = Schema Module

- CodeList_CurrencyCode_ISO_7_04
- CodeList_LanguageCode_ISO_7_04
- CodeList_MIMEMediaTypeCode_IANA_7_04
- CodeList_UnitCode_UNECE_7_04

- UnqualifiedDataTypes-NewATG2
- SI-Enterprise-QualifiedDataTypes
  - SI-Enterprise-CommonBasicComponents
  - SI-Enterprise-CommonAggregateComponents
  - SI-Enterprise-InventoryAvailabilityRequest
  - SI-Enterprise-InventoryAvailabilityResponse
Populate Document Schemas

- Populate ‘Document’ Schemas
  - Create Message Assembly
  - Fill out ABIE / BBIE / ASBIE content
  - Create Aggregate Schema
  - Create Basic Schema
  - Define constraints (cardinality, optionality)
  - Create documentation
    (some documentation omitted from examples for brevity)

```xml
<xsd:schema>
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
</xsd:schema>
```
Populate Document Schemas

- Populate ‘Document’ Schemas
  - Create Message Assembly

### Inventory Availability - Data Analysis Worksheet

<table>
<thead>
<tr>
<th>XML Tag Name</th>
<th>XML ComplexType</th>
<th>CCTS Dictionary Entry Name</th>
<th>ABIE/ABIE</th>
<th>Object Class Qualifier</th>
<th>Object Class Term</th>
<th>Property Term</th>
<th>Property Term</th>
<th>Represntatio</th>
</tr>
</thead>
<tbody>
<tr>
<td>InventoryAvailability Request</td>
<td>InventoryAvailabilityRequestType</td>
<td>InventoryAvailability_Request. Details</td>
<td>ABIE</td>
<td>InventoryAvailability</td>
<td>Request</td>
<td>Details</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InventoryOrganization</td>
<td>InventoryOrganizationType</td>
<td>InventoryAvailability_Request. Inventory. Organization. Inventory. Organization</td>
<td>ASBIE</td>
<td>InventoryAvailability</td>
<td>Request</td>
<td>Inventory</td>
<td>Organization</td>
<td></td>
</tr>
<tr>
<td>InventoryItem</td>
<td>InventoryItemType</td>
<td>InventoryAvailability_Request. Inventory. Inventory. Item</td>
<td>ASBIE</td>
<td>InventoryAvailability</td>
<td>Request</td>
<td>Inventory</td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>SupplierOrganization</td>
<td>SupplierOrganizationType</td>
<td>InventoryAvailability_Request. Supplier. Organization. Supplier. Organization</td>
<td>ASBIE</td>
<td>InventoryAvailability</td>
<td>Request</td>
<td>Supplier</td>
<td>Organization</td>
<td></td>
</tr>
<tr>
<td>InventoryAvailability Response</td>
<td>InventoryAvailabilityResponseType</td>
<td>InventoryAvailability_Response. Details</td>
<td>ABIE</td>
<td>InventoryAvailability</td>
<td>Response</td>
<td>Details</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SupplierOrganization</td>
<td>SupplierOrganizationType</td>
<td>InventoryAvailability_Response. Supplier. Organization. Supplier. Organization</td>
<td>ASBIE</td>
<td>InventoryAvailability</td>
<td>Response</td>
<td>Supplier</td>
<td>Organization</td>
<td></td>
</tr>
<tr>
<td>InventoryItem</td>
<td>InventoryItemType</td>
<td>InventoryAvailability_Response. Inventory. Inventory. Item</td>
<td>ASBIE</td>
<td>InventoryAvailability</td>
<td>Response</td>
<td>Inventory</td>
<td>Item</td>
<td></td>
</tr>
</tbody>
</table>

**remember these 2 ABIEs...**
Populate Document Schemas

- Create Message Assembly

```xml
<xsd:schema ...
<xsd:element name="InventoryAvailabilityRequest" type="InventoryAvailabilityRequestType">
  <xsd:complexType name="InventoryAvailabilityRequestType">
    <xsd:annotation>
      <xsd:documentation>
        <ccts:CategoryCode>ABIE</ccts:CategoryCode>
        <ccts:DictionaryEntryName/>
        <ccts:Definition>This holds...</ccts:Definition>
        <ccts:ObjectClass/>
        <ccts:RepresentationTerm/>
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref="cac:InventoryOrganization"/>
      <xsd:element ref="cac:InventoryItem" maxOccurs="unbounded"/>
      <xsd:element ref="cac:SupplierOrganization"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

ABIE
Populate Document Schemas

CodeList_CurrencyCode_ISO_7_04

CodeList_LanguageCode_ISO_7_04

CodeList_MIMEMediaTypeCode_IANA_7_04

CodeList_UnitCode_UNECE_7_04

UnqualifiedDataTypes-NewATG2

Enterprise-QualifiedDataTypes

CommonBasicComponents

Document

<xsd:element name="InventoryAvailabilityRequest"

InventoryAvailabilityResponse
Document & Common Schemas

- Document Schemas ‘use’ the content defined in the common (or enterprise) Schemas
- Those Schemas will be used by any number of Document Schema modules
Customizing UBL

• Guidelines For The Customization of UBL v1.0
  – Customization will happen
  – It will be done by a wide range of users
  – Changes will be driven by real world needs
  – These needs will be expressed as context drivers
UBL Derivation - Conformant

- UBL core
- Industry implementation
- User-specific implementation
UBL Derivation – Conformant and Non-conformant
Limits on the Application of Context
Thank You