Break Time
UBL Specification

Data Types
Relationship of Data Types

• Representation Term:
  - The value domain.
  - Conceptual.

• Core Component Type:
  - Still conceptual.
  - The actual content…
  - plus one or more Supplementary Components.

• Data Type:
  - Defines the set of valid values.
  - Restrictions on the Core Component Type.
  - Maps to implementation syntax.
Relationship of Data Types

We are interested in these
UBL Data Types

- **Unqualified** Data Types:
  - Corresponding Data Type for each CCTS Core Component Type.
    - We have seen these earlier.
  - Defined by UN/CEFACT.

- **Qualified** Data Types:
  - Specialization of Unqualified Data Type.
    - Defined by UBL (for defining code lists).
  - Can be created/extended/customized by implementors.
    - Used for facets and value constraints.
ULB Qualified Data Types

- 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
- Packaging Type Code
- Payment Means Code
- Substitution Status Code
- Transportation Status Code
- Transport Equipment Type Code
- Transport Mode Code
- Unit Of Measure Code

• Nominated BBIEs must use these.
Implementing a UBL Qualified Data Type

• Tax Currency Code uses the Currency Code type.

```xml
<xsd:complexType name="TaxCurrencyCodeType">
  <xsd:simpleContent>
    <xsd:extension base="qdt:CurrencyCodeType"/>
  </xsd:simpleContent>
</xsd:complexType>
```
Currency Code Data Type

Type defined by UN/CEFACT

Values are based on standard codes

Document with actual values (genricode format)
Fragment of genericode File (for ISO 4217 Alpha-3)

```xml
<Row>
  <Value ColumnRef="code">
    <SimpleValue>ARS</SimpleValue>
  </Value>
  <Value ColumnRef="name">
    <SimpleValue>Argentine Peso</SimpleValue>
  </Value>
</Row>

<Row>
  <Value ColumnRef="code">
    <SimpleValue>AUD</SimpleValue>
  </Value>
  <Value ColumnRef="name">
    <SimpleValue>Australian Dollar</SimpleValue>
  </Value>
</Row>
```

These terms can be changed/extended.
genericode is...

• A customizable XML document format for listing any sets of values.
  • Not just for codes
• An OASIS Technical Specification
  • in progress.
• Used by UBL to define code list values.
• An alternative to using enumerated lists.
genericode is not...

- A normative part of the UBL standard.
  - Cannot be verified, because...
- A validation tool.
  - Only documents values.
  - But can be used in a validation process.
UBL Specification
Physical Information Models
Schema Representations

• Different schema languages.
• Model can vary along many dimensions:
  - Type definitions
  - Scoping of elements, attributes, and types
  - Use of namespaces
• The document instance might look identical in all cases.
• UBL has rules for schema representation:
  - Known as “Naming and Design Rules (NDR)”
Example of a UBL Schema

```
<xsd:element ref="cbc:CustomerAssignedAccountID" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:Component>
        <ccts:ComponentType>BBIE</ccts:ComponentType>
        <ccts:Definition>An identifier for the Customer's account, assigned by the Customer itself.</ccts:Definition>
        <ccts:Cardinality>0..1</ccts:Cardinality>
        <ccts:ObjectClass>Customer Party</ccts:ObjectClass>
        <ccts:PropertyTermQualifier>Customer Assigned</ccts:PropertyTermQualifier>
        <ccts:PropertyTerm>Account Identifier</ccts:PropertyTerm>
        <ccts:RepresentationTerm>Identifier</ccts:RepresentationTerm>
        <ccts:DataType>Identifier. Type</ccts:DataType>
      </ccts:Component>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```
Categories of NDR rules

- Attribute Declaration Rules (ATD)
- Code List Rules (CDL)
- ComplexType Definition Rules (CTD)
- ComplexType Naming Rules (CTN)
- Documentation Rules (DOC)
- Element Declaration Rules (ELD)
- Element Naming Rules (ELN)
- General Naming Rules (GNR)
- General Type Definition Rules (GTD)
- General XML Schema Rules (GXS)
- Modeling Constraints Rules (MDC)
- Naming Constraints Rules (NMC)
- Namespace Rules (NMS)
- Root Element Declaration Rules (RED)
- Schema Structure Modularity Rules (SSM)
- Versioning Rules (VER)
Some UBL NDRs

- \([G\text{NR2}]\) UBL XML element and type names MUST be consistently derived from CCTS conformant dictionary entry names.
Generating Schemas

- W3C XML Schema Language is complex:
  - Creating consistent schema from scratch is hard.
- So UBL generated schema from spreadsheets.
  - Based on UBL NDRs.
- Existing UBL generation tools are:
  - UBLer by Invinet Systems in Spain.
  - GEFEG.FX by GEFEG mbH in Germany.
  - UBLish by SoftML in Singapore.
- Emphasizes that designing models is more important than designing schema.
UBL Schema Creation

Modeling

Spreadsheets

Imported from UN/CEFACT

Based on UBL NDRs

Automated process

Schema module for Unqualified Datatypes

Schema modules for document types

Schema module for reusable ABIEs

Schema module for reusable BBIEs

Schema module for Qualified Datatypes
Case Study Scenario
Case Study Scenario

- IYT Corporation wants to order beeswax from Consortio.

But

- Consortio requires customization of UBL Party structure.
IYT Corporation (UBL 2.0)

<cac:BuyerCustomerParty>
<cbc:CustomerAssignedAccountID>XFB01</cbc:CustomerAssignedAccountID>
<cbc:SupplierAssignedAccountID>GT00978567</cbc:SupplierAssignedAccountID>
<cac:Party>
  <cac:PartyName>
    <cbc:Name>IYT Corporation</cbc:Name>
  </cac:PartyName>
  <cac:PostalAddress>
    <cbc:StreetName>Avon Way</cbc:StreetName>
    <cbc:BuildingName>Thereabouts</cbc:BuildingName>
    <cbc:BuildingNumber>56A</cbc:BuildingNumber>
    <cbc:CityName>Bridgtow</cbc:CityName>
    <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
    <cbc:CountrySubentity>Avon</cbc:CountrySubentity>
    <cac:AddressLine>
      <cbc:Line>3rd Floor, Room 5</cbc:Line>
    </cac:AddressLine>
  </cac:PostalAddress>
</cac:Party>
</cac:BuyerCustomerParty>
Customization Requirements

<cac:BuyerCustomerParty>
 <cbc:CustomerAssignedAccountID>XFB01</cbc:CustomerAssignedAccountID>
 <cbc:SupplierAssignedAccountID>GT00978567</cbc:SupplierAssignedAccountID>
 <cac:Party>
   <cac:PartyName>
     <cbc:Name>IYT Corporation</cbc:Name>
   </cac:PartyName>
 <cac:PostalAddress>
   <cbc:StreetName>Avon Way</cbc:StreetName>
   <cbc:BuildingName>Thereabouts</cbc:BuildingName>
   <cbc:BuildingNumber>56A</cbc:BuildingNumber>
   <cbc:CityName>Bridgtow</cbc:CityName>
   <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
   <cbc:CountrySubentity>Avon</cbc:CountrySubentity>
   <cac:AddressLine>
     <cbc:Line>3rd Floor, Room 5</cbc:Line>
   </cac:AddressLine>
   <cac:Country>
   </cac:Country>
 </cac:PostalAddress>
 </cac:Party>
</cac:BuyerCustomerParty>

Add an Account ID from trade consortium (Consip?)

Identify preferred name

Postal Zone has to be XXXX XXX format

Only allow an address with PO Box, Postal Zone and Country

Also add Post Office Name for PO Box.
Information Required

<:BuyerCustomerParty>
  <:TradeAssignedAccountID>AAA09382</TradeAssignedAccountID>
  <:Party>
    <:PreferredPartyName>
      <:Name>IYT Corporation</Name>
    </PreferredPartyName>
    <:PostalAddress>
      <:Postbox>1279</Postbox>
      <:Postoffice>Bridgtow</Postoffice>
      <:PostalZone>ZZ99 1ZZ</PostalZone>
      <:Country>
        <:IdentificationCode>GB</IdentificationCode>
      </Country>
    </PostalAddress>
  </Party>
</BuyerCustomerParty>

New BBIEs
New ASBIE
New ABIE
New Data Type
Customizing UBL

Conformance or Compatibility
• An XML document is considered UBL conformant if there are no constraint violations when validating it against a published UBL schema.
• We should only say “conformant” when we mean this.
UBL Compatibility

- Consistent or in keeping with the principles behind UBL's models and/or their development.
- Use UBL as a set of building blocks.
- We cannot ensure interoperability.
  - some degree of familiarity and re-use.
- Degrees of compatibility.
  - Like spoken languages.
- Towards interoperability.
  - Evolution not revolution.
  - “Steps up the ladder toward interoperability”.
There is value in implementers customizing in a consistent manner.  
- Potential re-use of customized patterns.

This involves a common approach to:
- **Design:**  
  • knowing what models to change and how
- **Specification:**  
  • communicating these new models (both to humans and applications)
- **Validation:**  
  • ensuring documents conform to the specification.
Customizing UBL

Designing Customizations
Typical Requirements

• Creating new BBIEs.
  - Developing completely new BBIE.
  - Qualifying the data type of existing BBIE.

• Assembling New ABIEs.
  - Omitting ASBIEs or BBIEs
  - Adding new ASBIEs or BBIEs
  - Modifying cardinality of ASBIEs or BBIEs.

• Creating new ASBIEs.

• Assembling new document types.

• Both restricted and extended customizations.
Design for Conformance

• All schema-valid instances of a conformant customization are simultaneously schema-valid instances of UBL.
  – But not the other way around.
• Conformant designs only allow for restrictions:
  – Subsets of the document model.
  – Constraints on document content.
Subsets of Document Model

- 800,000 potential elements within an Order document.
- Subsets remove any unnecessary optional information entities.
- Cannot reduce the permitted minimum number of occurrences.
- Cannot extend the permitted maximum number of occurrences.
  - 0..1 can become 1..1, or 0..0 (not used)
  - 0..n can become 0..1, 1..n, or 0..0 (not used)
  - 1..n can become 1..1
  - 1..1 cannot be restricted
Designing for Compatibility

• All schema-valid instances of a compatible customization are not schema-valid instances of UBL.
  – But may be the other way around.
• Compatible design also allows for extensions (supersets)
  – Adding to the model any information entities that are needed to satisfy business requirements.
Criteria for Compatibility

- Re-use UBL Patterns
  - Re-use Information Entities.
  - Data Types.

- Use UBL principles for new Information Entities:
  - Normalize aggregates.
  - Base on conceptual model.
  - Design using patterns.
  - Use ebXML CCTS.
  - Use UBL NDR for any schema.
A customized ABIE means creating a new ABIE.
- Customizing a BBIE creates a new ABIE
  - Customizing a Data Type creates a new BBIE
  - Customizing an ASBIE creates a new ABIE
- A customized ABIE means a new document type.
Standard UBL Document
Customized ABIE
Customized BBIE or Datatype
Creating New Document Types

- Assemble new pathways from customized component model.
- Follow principles for document assembly...
  - Choose entry point ABIE.
  - Be guided by business rules.
  - Assemble BBIEs...
    - Mandatory and optional
  - Assemble associations…
    - Mandatory and optional
  - Note cardinality and directional constraints.
  - Proceed recursively through other ABIEs.
Customizing UBL

Specifying Customizations
Specification Options

- W3C Schema derivation.
- Independent W3C Schemas.
  - By redefinition
  - By re-use
  - By using UBLExtension
- XPath files
  - Specify UBL restrictions.
Specify Independent Schemas

• Generate new schemas.
  - Edit a model representation and translate the model into a schema expression
    • As UBL itself did.
    - May restrict the customization possibilities.

• Create new schemas.
  - Edit an existing UBL schema
  - Adhering to NDRs

• Challenge is to maintain compatibility.
Customizing UBL

Specifying Restricted Customizations
Specifying Subset Models

- Model restriction involves some combination of
  - Restricting BBIEs and ASBIEs.
    - removing optional information entities.
  - Restricting Data Types
    - adding constraints such as a code list or a length facet.
- Customized documents are a subset.
  - Similar to EDI implementations.
- Restricted documents should be UBL conformant.
  - UBL SBS, OIOUBL and NES follow this approach.
Restriction Requirements

<cac:BuyerCustomerParty>
<cbc:CustomerAssignedAccountID>XFB01</cbc:CustomerAssignedAccountID>
<cbc:SupplierAssignedAccountID>GT00978567</cbc:SupplierAssignedAccountID>
  <cac:Party>
    <cac:PartyName>
      <cbc:Name>IYT Corporation</cbc:Name>
    </cac:PartyName>
    <cac:PostalAddress>
      <cbc:StreetName>Avon Way</cbc:StreetName>
      <cbc:BuildingName>Thereabouts</cbc:BuildingName>
      <cbc:BuildingNumber>56A</cbc:BuildingNumber>
      <cbc:CityName>Bridgtow</cbc:CityName>
      <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
      <cbc:CountrySubentity>Avon</cbc:CountrySubentity>
      <cac:AddressLine>
        <cbc:Line>3rd Floor, Room 5</cbc:Line>
      </cac:AddressLine>
    </cac:PostalAddress>
    <cac:Country>
    </cac:Country>
  </cac:Party>
</cac:BuyerCustomerParty>

Postal Zone has to be XXXX XXX format

Only allow an address with PO Box and Post Office
Restricted Model

```
«ABIE»
Address

# «BBIE» ID: Identifier [0..1]
- «BBIE» Address Type Code: Code [0..1]
- «BBIE» Address Format Code: Code [0..1]
# «BBIE» Postbox: Text [0..1]
# «BBIE» Floor: Text [0..1]
# «BBIE» Room: Text [0..1]
# «BBIE» Street Name: Name [0..1]
# «BBIE» Additional Street Name: Name [0..1]
- «BBIE» Block Name: Name [0..1]
# «BBIE» Building Name: Name [0..1]
# «BBIE» Building Number: Text [0..1]
# «BBIE» Inhouse Mail: Text [0..1]
# «BBIE» Department: Text [0..1]
- «BBIE» Mark Attention: Text [0..1]
- «BBIE» Mark Care: Text [0..1]
- «BBIE» Plot Identification: Text [0..1]
- «BBIE» City Subdivision Name: Name [0..1]
# «BBIE» City Name: Name [0..1]
# «BBIE» Postal Zone: Text [0..1]
# «BBIE» Country Subentity: Text [0..1]
# «BBIE» Country Subentity Code: Code [0..1]
# «BBIE» Region: Text [0..1]
# «BBIE» District: Text [0..1]
# «BBIE» Timezone Offset: Text [0..1]
```

subset

tightened constraints

restriction on format
Specifying Constraints on Content

• Implemented as Qualified Data Types:
  – Refinements of:
    • Unqualified Data Types.
    • Other Qualified Data Types.
  
• Use genericode format for defining values!
Example of NES Restriction
A lower level library cannot extend cardinality or add BIEs that are not part of the library directly above.
• The UBL 2.0 Small Business Subset is one example of how a conformant subset may be specified through the use of a subset schema
  – All instances of UBL 2.0 SBS are instances of full UBL 2.0
  – The UBL 2.0 SBS schemas are pruned copies of the normative UBL 2.0 schemas

• Research is underway to demonstrate the pruned schemas are provably correct proper subsets of the full schemas
Customizing UBL

Specifying Extension Customizations
Extending UBL

- The UBL Library is 80/20.
  - Generic context of use, not specific.
- Extension involves some combination of:
  - Creating new ABIEs, BBIEs and ASBIEs.
  - Extending cardinality
    - Making optional, mandatory.
  - Creating new Unqualified Data Types.
    - Not derived from existing Data Types.
- Extended documents are a superset.
  - Should be UBL compatible.
    - CODICE followed this approach.
Customizing UBL

Extension using UBLExtension
• Ensures conformance.
  - All instances will pass UBL schema validation.
• No inherent constraints.
  - Based on xsd:any.
• Use if nothing else fits.
  - Include legacy document formats.
• The "black hole" of UBL.
  - Use with caution.
• Needs care when implementing.
UBLExtension Caveats

• Should be documented.
  – Specifying context.

• Should be registered.
  – Somewhere?

• Should never be used for information that may properly be conveyed in standard UBL patterns elsewhere in the document.

• Extension content model should aim for UBL compatibility.
  – Conformance is not enough.
UBLEExtension Structure
UBLExtension Sample

```
<Order>
  <ext:UBLExtension>
    <ext:ExtensionContent>
      <eac:BuyerExtendedCustomerParty>
        <ebc:TradeAssignedAccountID>T123356</ebc:TradeAssignedAccountID>
        <eac:ExtendedParty>
          <eac:PrimaryPartyName>
            <cbc:Name>IYT</cbc:Name>
          </eac:PrimaryPartyName>
          <eac:AlternativePostalExtendedAddress>
            <ebc:Postoffice>South Bridgtow</ebc:Postoffice>
            <eac:Address>
              <cbc:Postbox>2234</cbc:Postbox>
              <cbc:PostalZone>ZZ99 0AA</cbc:PostalZone>
              <cac:Country>
              </cac:Country>
            </eac:Address>
          </eac:AlternativePostalExtendedAddress>
        </eac:ExtendedParty>
      </eac:BuyerExtendedCustomerParty>
    </ext:ExtensionContent>
  </ext:UBLExtension>
</Order>

Can be anything in here

Normal UBL Order

… standard order details …
```
OIOUBL UBLExtension Example

```xml
<ext:UBLExtensions>
  <ext:UBLExtension>
    <cbc:ID>WMP1</cbc:ID>
    <cbc:Name>WMData</cbc:Name>
    <ext:ExtensionAgencyID>EAI1</ext:ExtensionAgencyID>
    <ext:ExtensionAgencyName>EAN1</ext:ExtensionAgencyName>
    <ext:ExtensionAgencyURI>EAU1</ext:ExtensionAgencyURI>
    <ext:ExtensionURI>urn:wmdata.dk:example</ext:ExtensionURI>
    <ext:ExtensionReason>wmdata legacy invoice material</ext:ExtensionReason>
    <ext:ExtensionContent>
      <wmp:LegacyExtension
        xmlns:wmp="urn:urn:wmdata.dk:example">
        ...legacy invoice stuff...
      </wmp:LegacyExtension>
    </ext:ExtensionContent>
  </ext:UBLExtension>
</ext:UBLExtensions>
```

Registration details

A common application
• Identifies the legitimate elements and attributes in a document instance.
• Document definition:
  – List of all legitimate XPaths.
  – Can be very large!
• They can be generated from:
  – XSD schemas.
  – XML instances.
  – UBL spreadsheets.
• Used by UBL.
  – Small Business Subset definitions
Sample XPath Specification

```
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/16 1..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:StreetName16.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:BuildingName17.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:BuildingNumber18.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:CitySubdivisionName19.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:CityName20.1 0..1
```
Validating UBL Customizations
Validating XML Documents

• Model/Profile validation.
• Content/Value validation.
• W3C Schemas.
  - Parser applications.
• Other schema languages.
  - Relax NG, ASN.1
• Programmable languages.
  - ISO Schematron (and stylesheets).
  - Coded applications (scripts/programs).
Identifying Customizations

• Profiles enable 'families' of customizations.
  – “Stand Alone Invoicing” is a profile for NES.
    • Meaning the requirements for stand alone invoicing in the northern European context.

• Special BBIEs allow instances to identify their customization specifications:
  • 'UBLVersionID'
    • the UBL version being customized.
  • 'UBLCustomizationID'
    • an identifier (such as a namespace) for a specified customization of UBL.
  • 'UBLProfileID'
    • an identifier (such as a namespace) for a specified profile of the customization being used.
UBL Schema Validation

- A document schema defines all rules for validation.
- Documents must pass this to be deemed UBL conformant.
- Degree of acceptability depends on number of business rules tested.
  - Limited with W3C Schema
Validating UBL Conformance
Validating Using XPath

• XPaths can be utilized to validate elements.
• Can implement an exhaustive proof of conformance and/or compatibility.
• Conformance, by programmatically checking:
  – all of the XPath entries of the subset are found in UBL.
  – none of the mandatory items in the UBL are missing from the subset.
• Extensions, by programmatically checking:
  – none of the mandatory items in the UBL are missing from the superset.
Multi-Stage Validation

- Pre-conformance processes
  - Separate extensions.
- Conformance processes
  - Validate against UBL Schema
- Post-conformance processes
  - Validate restrictions.
- Experience suggests we will evolve toward this approach.
Validating Customizations

Extension XPaths

UBL Schema

Restricted XPaths

Any Document

UBL Portion of Document

UBL Conformant Document

Valid Customized Document

© Document Engineering Services 2008
UBLExtension and Stub

```
<Order>
  <ext:UBLExtension>
    <ext:ExtensionContent>
      <eac:BuyerExtendedCustomerParty>
        <eac:PrimaryPartyName>
          <cbc:Name>IYT</cbc:Name>
        </eac:PrimaryPartyName>
        <eac:AlternativePostalExtendedAddress>
          <ebc:Postoffice>South Bridgtow</ebc:Postoffice>
          <eac:Address>
            <cbc:Postbox>2234</cbc:Postbox>
            <cbc:PostalZone>ZZ99 0AA</cbc:PostalZone>
            <cac:Country>
            </cac:Country>
          </eac:Address>
        </eac:AlternativePostalExtendedAddress>
        <eac:ExtendedParty>
          <eac:PrimaryPartyName>
            <cbc:Name>T123356</cbc:Name>
          </eac:PrimaryPartyName>
          <eac:AlternativePostalExtendedAddress>
            <ebc:Postoffice>South Bridgtow</ebc:Postoffice>
            <eac:Address>
              <cbc:Postbox>2234</cbc:Postbox>
              <cbc:PostalZone>ZZ99 0AA</cbc:PostalZone>
              <cac:Country>
              </cac:Country>
            </eac:Address>
          </eac:AlternativePostalExtendedAddress>
        </eac:ExtendedParty>
      </eac:BuyerExtendedCustomerParty>
    </ext:ExtensionContent>
  </ext:UBLExtension>
  <cac:BuyerCustomerParty>
    <cac:Party>
      <cac:PostalAddress>
        <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
        <cac:Country>
        </cac:Country>
      </cac:PostalAddress>
    </cac:Party>
  </cac:BuyerCustomerParty>
</Order>
```
Customized Document Architecture

• Use standard UBL document schema.
  – UBLExtension contains entire document to be processed.
  – Document body contains only the valid UBL component.
    • This “stub” is validated by UBL schema.

• Data is duplicated.
  – Is this a problem?

• Always UBL conformant.
  – Consistent common data.
Summary and Feedback
Summary

- Background to Document Design
- UBL Design
- UBL Specification
- Designing Customizations
- Specifying Customizations
- Validating Customizations
Thank You
Break Time
UBL Specification
Data Types
Relationship of Data Types

- **Representation Term:**
  - The value domain.
  - Conceptual.
- **Core Component Type:**
  - Still conceptual.
  - The actual content...
  - plus one or more Supplementary Components.
- **Data Type:**
  - Defines the set of valid values.
  - Restrictions on the Core Component Type.
  - Maps to implementation syntax.
We are interested in these
UBL Data Types

- **Unqualified** Data Types:
  - Corresponding Data Type for each CCTS Core Component Type.
  - We have seen these earlier.
  - Defined by UN/CEFACT.

- **Qualified** Data Types:
  - Specialization of Unqualified Data Type.
    - Defined by UBL (for defining code lists).
    - Can be created/extended/customized by implementors.
    - Used for facets and value constraints.
<table>
<thead>
<tr>
<th>UBL Qualified Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Nominated BBIEs must use these.</td>
</tr>
<tr>
<td>• Allowance Charge Reason Code   • Longitude Direction Code</td>
</tr>
<tr>
<td>• Channel Code             • Operator Code</td>
</tr>
<tr>
<td>• Chip Code                • Packaging Type Code</td>
</tr>
<tr>
<td>• Country Identification Code • Payment Means Code</td>
</tr>
<tr>
<td>• Currency Code            • Substitution Status Code</td>
</tr>
<tr>
<td>• Document Status Code     • Transportation Status Code</td>
</tr>
<tr>
<td>• Latitude Direction Code  • Transport Equipment Type Code</td>
</tr>
<tr>
<td>• Line Status Code         • Transport Mode Code</td>
</tr>
<tr>
<td>• Unit Of Measure Code</td>
</tr>
</tbody>
</table>
• Tax Currency Code uses the Currency Code type.

```xml
<xsd:complexType name="TaxCurrencyCodeType">
  <xsd:simpleContent>
    <xsd:extension base="qdt:CurrencyCodeType"/>
  </xsd:simpleContent>
</xsd:complexType>
```
Currency Code Data Type

```xml
<xsd:complexType name="CurrencyCodeType">
  <xsd:simpleContent>
    <xsd:restriction base="xsd:CodeType">
      <xsd:attribute name="listID" type="xsd:normalizedString" use="optional" default="ISO 4217 Alpha"/>
      <xsd:attribute name="listAgencyID" type="xsd:normalizedString" use="optional" default="6"/>
      <xsd:attribute name="listAgencyName" type="xsd:string" use="optional" default="United Nations Economic Commission for Europe"/>
      <xsd:attribute name="listVersionID" type="xsd:normalizedString" use="optional" default="2001"/>
      <xsd:attribute name="name" type="xsd:string" use="optional"/>
      <xsd:attribute name="languageID" type="xsd:language" use="optional" default="en"/>
      <xsd:attribute name="listURI" type="xsd:anyURI" use="optional" default="http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/un:cefact/CurrencyCode-2.0.gc"/>
      <xsd:attribute name="listSchemeURI" type="xsd:anyURI" use="optional" default="urn:un:unece:uncefact:codelist:specification:54217:2001"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
```

Type defined by UN/CEFACT

Values are based on standard codes

Document with actual values (genericode format)
Fragment of genericode File (for ISO 4217 Alpha-3)

<table>
<thead>
<tr>
<th>Code value</th>
<th>Name value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARS</td>
<td>Argentine Peso</td>
</tr>
<tr>
<td>AUD</td>
<td>Australian Dollar</td>
</tr>
</tbody>
</table>
genericode is...

- A customizable XML document format for listing any sets of values.
  - Not just for codes
  - An OASIS Technical Specification
    - in progress.
  - Used by UBL to define code list values.
  - An alternative to using enumerated lists.
genericode is not...

- A normative part of the UBL standard.
  - Cannot be verified, because...
- A validation tool.
  - Only documents values.
  - But can be used in a validation process.
UBL Specification
Physical Information Models
Schema Representations

- Different schema languages.
- Model can vary along many dimensions:
  - Type definitions
  - Scoping of elements, attributes, and types
  - Use of namespaces
- The document instance might look identical in all cases.
- UBL has rules for schema representation:
  - Known as "Naming and Design Rules (NDR)"

[Show an instance example]
Example of a UBL Schema

```xml
<xsd:element ref="cbc:CustomerAssignedAccountID" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:Component>
        <ccts:ComponentType>BBIE</ccts:ComponentType>
        <ccts:DictionaryEntryName>Customer Party, Customer Assigned_ Account Identifier</ccts:DictionaryEntryName>
        <ccts:Definition>An identifier for the Customer's account, assigned by the Customer itself.</ccts:Definition>
        <ccts:ObjectClass>Customer Party</ccts:ObjectClass>
        <ccts:PropertyTermQualifier>Customer Assigned</ccts:PropertyTermQualifier>
        <ccts:PropertyTerm>Account Identifier</ccts:PropertyTerm>
        <ccts:RepresentationTerm>Identifier</ccts:RepresentationTerm>
        <ccts:DataType>Identifier</ccts:DataType>
      </ccts:Component>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

Element Tag name
Documentation (not validated)
CCTS metadata
Categories of NDR rules

- Attribute Declaration Rules (ATD)
- Code List Rules (CDL)
- Complex Type Definition Rules (CTD)
- Complex Type Naming Rules (CTN)
- Documentation Rules (DOC)
- Element Declaration Rules (ELD)
- Element Naming Rules (ELN)
- General Naming Rules (GNR)
- General Type Definition Rules (GTD)
- General XML Schema Rules (GXS)
- Modeling Constraints Rules (MDC)
- Naming Constraints Rules (NMC)
- Namespace Rules (NMS)
- Root Element Declaration Rules (RED)
- Schema Structure Modularity Rules (SSM)
- Versioning Rules (VER)
Some UBL NDRs

[GNR2] UBL XML element and type names MUST be consistently derived from CCTS conformant dictionary entry names.

<table>
<thead>
<tr>
<th>UBL Name</th>
<th>Dictionary Entry Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address ID</td>
<td>Address, Details</td>
</tr>
<tr>
<td>Address Type Code</td>
<td>Address, Type Code, Code</td>
</tr>
<tr>
<td>Address Format Code</td>
<td>Address, Format Code, Code</td>
</tr>
<tr>
<td>Postbox</td>
<td>Address, Postbox, Text</td>
</tr>
<tr>
<td>Room</td>
<td>Address, Room, Text</td>
</tr>
<tr>
<td>StreetName</td>
<td>Address, Street Name</td>
</tr>
<tr>
<td>AdditionalStreetName</td>
<td>Address, Additional, Street Name, Name</td>
</tr>
</tbody>
</table>
Generating Schemas

- W3C XML Schema Language is complex:
  - Creating consistent schema from scratch is hard.
- So UBL generated schema from spreadsheets.
  - Based on UBL NDRs.
- Existing UBL generation tools are:
  - UBLer by Invinet Systems in Spain.
  - GEFEG.FX by GEFEG mbH in Germany.
  - UBLish by SoftML in Singapore.
- Emphasizes that designing models is more important than designing schema.
UBL Schema Creation

- Modeling
- Spreadsheets
- Automated process
- Imported from UN/CEFACT

- Schema module for document types
- Schema module for reusable ABIEs
- Schema module for reuseable BBIEs
- Schema module for Qualified Datatypes

Based on UBL NDTRs

Modeling

Spreadsheet

Automated process

Imported from UN/CEFACT

Schema module for Unqualified Datatypes
Case Study Scenario
Case Study Scenario

- IYT Corporation wants to order beeswax from Consortio.
  But
- Consortio requires customization of UBL Party structure.
IYT Corporation (UBL 2.0)

```
<cac:BuyerCustomerParty>
  <cbc:CustomerAssignedAccountID>XFB01</cbc:CustomerAssignedAccountID>
  <cbc:SupplierAssignedAccountID>GT00978567</cbc:SupplierAssignedAccountID>
  <cac:Party>
    <cac:PartyName>
      <cbc:Name>IYT Corporation</cbc:Name>
    </cac:PartyName>
    <cac:PostalAddress>
      <cbc:StreetName>Avon Way</cbc:StreetName>
      <cbc:BuildingName>Thereabouts</cbc:BuildingName>
      <cbc:BuildingNumber>56A</cbc:BuildingNumber>
      <cbc:CityName>Bridgtow</cbc:CityName>
      <cbc:PostalZone>ZZ99 1ZZ</cbc:PostalZone>
      <cac:AddressLine>3rd Floor, Room 5</cac:AddressLine>
      <cac:Country>
      </cac:Country>
    </cac:PostalAddress>
  </cac:Party>
</cac:BuyerCustomerParty>
```
Customization Requirements

- Add an Account ID from trade consortium (Consip?)
- Identify preferred name
- Postal Zone has to be XXXX XXXX format
- Only allow an address with PO Box, Postal Zone and Country
- Also add Post Office Name for PO Box.
<BuyerCustomerParty>
  <TradeAssignedAccountID>AAA9382</TradeAssignedAccountID>
  <Party>
    <PreferredPartyName>
      <Name>IYT Corporation</Name>
    </PreferredPartyName>
    <PostalAddress>
      <Postbox>1279</Postbox>
      <Postoffice>Bridgtow</Postoffice>
      <PostalZone>ZZ99 1ZZ</PostalZone>
      <Country>
        <IdentificationCode>GB</IdentificationCode>
      </Country>
    </PostalAddress>
  </Party>
</BuyerCustomerParty>

New BBIEs
New ASBIE
New ABIE
New Data Type
Customizing UBL
Conformance or Compatibility
• An XML document is considered UBL conformant if there are no constraint violations when validating it against a published UBL schema.
• We should only say “conformant” when we mean this.
UBL Compatibility

- Consistent or in keeping with the principles behind UBL's models and/or their development.
- Use UBL as a set of building blocks.
- We cannot ensure interoperability.
  - some degree of familiarity and re-use.
- Degrees of compatibility.
  - Like spoken languages.
- Towards interoperability.
  - Evolution not revolution.
  - “Steps up the ladder toward interoperability”.

© Document Engineering Services 2008
Consistent Customization

- There is value in implementers customizing in a consistent manner.
  - Potential re-use of customized patterns.
- This involves a common approach to:
  - **Design:**
    - knowing what models to change and how
  - **Specification:**
    - communicating these new models (both to humans and applications)
  - **Validation:**
    - ensuring documents conform to the specification.
Customizing UBL
Designing Customizations
Typical Requirements

- Creating new BBIEs.
  - Developing completely new BBIE.
  - Qualifying the data type of existing BBIE.
- Assembling New ABIEs.
  - Omitting ASBIEs or BBIEs
  - Adding new ASBIEs or BBIEs
  - Modifying cardinality of ASBIEs or BBIEs.
- Creating new ASBIEs.
- Assembling new document types.
- Both restricted and extended customizations.
Design for Conformance

- All schema-valid instances of a conformant customization are simultaneously schema-valid instances of UBL.
  - But not the other way around.
- Conformant designs only allow for restrictions:
  - Subsets of the document model.
  - Constraints on document content.
Subsets of Document Model

- 800,000 potential elements within an Order document.
- Subsets remove any unnecessary optional information entities.
- Cannot reduce the permitted minimum number of occurrences.
- Cannot extend the permitted maximum number of occurrences.
  - 0..1 can become 1..1, or 0..0 (not used)
  - 0..n can become 0..1, 1..n, or 0..0 (not used)
  - 1..n can become 1..1
  - 1..1 cannot be restricted
Designing for Compatibility

- All schema-valid instances of a compatible customization are not schema-valid instances of UBL.
  - But may be the other way around.
- Compatible design also allows for extensions (supersets)
  - Adding to the model any information entities that are needed to satisfy business requirements.
Criteria for Compatibility

- Re-use UBL Patterns
  - Re-use Information Entities.
  - Data Types.
- Use UBL principles for new Information Entities:
  - Normalize aggregates.
  - Base on conceptual model.
  - Design using patterns.
  - Use ebXML C茨TS.
  - Use UBL NDR for any schema.
The Customization Ripple Effect

- A customized ABIE means creating a new ABIE.
  - Customizing a BBIE creates a new ABIE
    - Customizing a Data Type creates a new BBIE
    - Customizing an ASBIE creates a new ABIE
- A customized ABIE means a new document type.
Customized ABIE

Diagram of Customized ABIE structure.
Customized BBIE or Datatype
Creating New Document Types

- Assemble new pathways from customized component model.
- Follow principles for document assembly...
  - Choose entry point ABIE.
  - Be guided by business rules.
  - Assemble BBIEs...
    • Mandatory and optional
  - Assemble associations...
    • Mandatory and optional
  - Note cardinality and directional constraints.
  - Proceed recursively through other ABIEs.
Customizing UBL
Specifying Customizations
Specification Options

- W3C Schema derivation.
- Independent W3C Schemas.
  - By redefinition
  - By re-use
  - By using UBLExtension
- XPath files
  - Specify UBL restrictions.
Specify Independent Schemas

- Generate new schemas.
  - Edit a model representation and translate the model into a schema expression
    - As UBL itself did.
    - May restrict the customization possibilities.

- Create new schemas.
  - Edit an existing UBL schema
  - Adhering to NDRs

- Challenge is to maintain compatibility.
Customizing UBL
Specifying Restricted Customizations
Specifying Subset Models

• Model restriction involves some combination of
  • Restricting BBIEs and ASBIEs.
    • removing optional information entities.
  • Restricting Data Types
    • adding constraints such as a code list or a length facet.
• Customized documents are a subset.
  • Similar to EDI implementations.
• Restricted documents should be UBL conformant.
  • UBL SBS, OIOUBL and NES follow this approach.
Restriction Requirements

Postal Zone has to be XXXX XXX format

Only allow an address with PO Box and Post Office
Restricted Model

- tightened constraints
- restriction on format
Specifying Constraints on Content

- Implemented as Qualified Data Types:
  - Refinements of:
    - Unqualified Data Types.
    - Other Qualified Data Types.
  - Use genericode format for defining values!
Example of NES Restriction

- C Delivery, Details
  - A Delivery, Identifier
  - X Delivery, Minimum Quantity, Quantity
  - X Delivery, Maximum Quantity, Quantity
  - A Delivery, Actual Delivery Date, Date
  - A Delivery, Actual Delivery Time, Time
  - A Delivery, Latest Delivery Date, Date
  - A Delivery, Latest Delivery Time, Time
  - X Delivery, Tracking Identifier, Identifier
  - C Delivery, Delivery Address, Address
  - C Delivery, Delivery Location, Location
  - C Delivery, Requested Delivery Period, Period
  - C Delivery, Promised Delivery Period, Period
  - C Delivery, Estimated Delivery Period, Period
  - C Delivery, Delivery Party, Party
  - X Delivery, Despatch
A lower level library cannot extend cardinality or add BIEs that are not part of the library directly above.
Subset Schema

- The UBL 2.0 Small Business Subset is one example of how a conformant subset may be specified through the use of a subset schema
  - All instances of UBL 2.0 SBS are instances of full UBL 2.0
  - The UBL 2.0 SBS schemas are pruned copies of the normative UBL 2.0 schemas
- Research is underway to demonstrate the pruned schemas are provably correct proper subsets of the full schemas
Customizing UBL

Specifying Extension Customizations
Extending UBL

• The UBL Library is 80/20.
  - Generic context of use, not specific.
• Extension involves some combination of:
  - Creating new ABIEs, BBIEs and ASBIEs.
  - Extending cardinality
    • Making optional, mandatory.
  - Creating new Unqualified Data Types.
    • Not derived from existing Data Types.
• Extended documents are a superset.
  - Should be UBL compatible.
    • CODICE followed this approach.
Customizing UBL
Extension using UBLExtension
UBLExtension

- Ensures conformance.
  - All instances will pass UBL schema validation.
- No inherent constraints.
  - Based on xsd:any.
- Use if nothing else fits.
  - Include legacy document formats.
- The "black hole" of UBL.
  - Use with caution.
- Needs care when implementing.
UBLExtension Caveats

- Should be documented.
  - Specifying context.

- Should be registered.
  - Somewhere?

- Should never be used for information that may properly be conveyed in standard UBL patterns elsewhere in the document.

- Extension content model should aim for UBL compatibility.
  - Conformance is not enough.
UBLExtension Structure
UBLExtension Sample

Can be anything in here

Normal UBL Order

standard order details
OIOUBL UBLExtension Example

```xml
<ext:UBLExtensions>
  <ext:UBLExtension>
    <cbc:ID>WMP1</cbc:ID>
    <cbc:Name>WMData</cbc:Name>
    <ext:ExtensionAgencyID>EAI1</ext:ExtensionAgencyID>
    <ext:ExtensionAgencyName>EAN1</ext:ExtensionAgencyName>
    <ext:ExtensionAgencyURI>EAU1</ext:ExtensionAgencyURI>
    <ext:ExtensionURI>urn:wmdata.dk:example</ext:ExtensionURI>
    <ext:ExtensionReason>wmdata legacy invoice material</ext:ExtensionReason>
    <ext:ExtensionContent>
      <wmp:LegacyExtension
        xmlns:wmp="urn:wmdata.dk:example">
        ...legacy invoice stuff...
      </wmp:LegacyExtension>
    </ext:ExtensionContent>
  </ext:UBLExtension>
</ext:UBLExtensions>
```

Registration details

A common application
XPath Specification

- Identifies the legitimate elements and attributes in a document instance.
- Document definition:
  - List of all legitimate XPaths.
  - Can be very large!
- They can be generated from:
  - XSD schemas.
  - XML instances.
  - UBL spreadsheets.
- Used by UBL:
  - Small Business Subset definitions
Sample XPath Specification

```xml
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/16   1..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:StreetName16.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:BuildingName17.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:BuildingNumber18.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:CitySubdivisionName19.1 0..1
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:Address/cbc:CityName20.1 0..1
```
Validating UBL Customizations
Validating XML Documents

- Model/Profile validation.
- Content/Value validation.
- W3C Schemas.
  - Parser applications.
- Other schema languages.
  - Relax NG, ASN.1
- Programmable languages.
  - ISO Schematron (and stylesheets).
  - Coded applications (scripts/programs).
Identifying Customizations

- Profiles enable 'families' of customizations.
  - "Stand Alone Invoicing" is a profile for NES.
    - Meaning the requirements for stand alone invoicing in the northern European context.
- Special BBIEs allow instances to identify their customization specifications:
  - 'UBLVersionID'
    - the UBL version being customized.
  - 'UBLCustomizationID'
    - an identifier (such as a namespace) for a specified customization of UBL.
  - 'UBLProfileID'
    - an identifier (such as a namespace) for a specified profile of the customization being used.
UBL Schema Validation

- A document schema defines all rules for validation.
- Documents must pass this to be deemed UBL conformant.
- Degree of acceptability depends on number of business rules tested.
  - Limited with W3C Schema
Validating UBL Conformance

Any Document

UBL Schema

UBL Conformant Document

© Document Engineering Services 2008
Validating Using XPath

- XPaths can be utilized to validate elements.
- Can implement an exhaustive proof of conformance and/or compatibility.
- Conformance, by programmatically checking:
  - all of the XPath entries of the subset are found in UBL.
  - none of the mandatory items in the UBL are missing from the subset.
- Extensions, by programmatically checking:
  - none of the mandatory items in the UBL are missing from the superset.
Multi-Stage Validation

- Pre-conformance processes
  - Separate extensions.
- Conformance processes
  - Validate against UBL Schema
- Post-conformance processes
  - Validate restrictions.
- Experience suggests we will evolve toward this approach.
Validating Customizations

- Any Document
  - Extension XPaths
  - UBL Portion of Document
  - UBL Schema
  - Restricted XPaths
  - Valid Customized Document

UBL Portion of Document

Conformant Document

Valid Document

Customized Document
UBLExtension and Stub

Document with extensions

Document without extensions
Customized Document Architecture

- Use standard UBL document schema.
  - UBLExtension contains entire document to be processed.
  - Document body contains only the valid UBL component.
    - This “stub” is validated by UBL schema.

- Data is duplicated.
  - Is this a problem?

- Always UBL conformant.
  - Consistent common data.
Summary and Feedback
Summary

- Background to Document Design
- UBL Design
- UBL Specification
- Designing Customizations
- Specifying Customizations
- Validating Customizations
Thank You