# UN/CEFACT
# XML Naming and Design Rules
# Version 3.0

## 1st Public Review
## 7 August 2008

# Abstract

This XML Naming and Design Rules specification defines an architecture and set of rules necessary to define, describe and use XML to consistently express business information exchanges. It is based on the World Wide Web consortium suite of XML specifications and the UN/CEFACT Core Components Technical Specification. This specification will be used by UN/CEFACT to define XML Schema and Schema documents which will be published and UN/CEFACT standards. It will also be used by other Standards Development Organizations who are interested in maximizing inter- and intra-industry interoperability.

# Table of Contents

# 1 Status of This Document

This UN/CEFACT technical specification is being developed in accordance with the UN/CEFACT/TRADE/R.650/Rev.4/Add.1/Rev.1 Open Development Process (ODP) for technical specifications. The UN/CEFACT Applied Technology Group (ATG) has approved it for broad public review.

This technical specification contains information to guide in interpretation or implementation.

Specification formatting is based on the Internet Society's Standard RFC format.

Distribution of this document is unlimited.

This version: UN/CEFACT XML Naming and Design Rules, Version 3.0 1st Public Review of 7 April 2008

Previous version: UN/CEFACT XML Naming and Design Rules, Version 3.0 ODP 5 Draft ATG Review 2 of 23 July 2008

This document may also be available in these non-normative formats: XML, XHTML with visible change markup. See also translations.

# 2 XML Naming and Design Rules Project Team Participants

We would like to recognize the following for their significant participation in the development of this technical specification.

**ATG2 Chair**

| | |
|---|---|
| Jostein Frømyr | EdiSys Consulting AS |

**Project Team Leader**

| | |
|---|---|
| Mark Crawford | SAP Labs LLC (U.S.) |

**Lead Editor**

| | |
|---|---|
| Michael Rowell | Oracle Corporation / OAGi |

**Contributors**

| | |
|---|---|
| Chuck Allen | HR-XML |
| Dipan Anarkat | GS1 |
| Serge Cayron | ACORD |
| Anthony Coates | Independent |
| David Connelly | OAGi |
| Mavis Cournane | Independent |
| Alain Dechamps | CEN |
| Michael Grimley | US Navy |
| Paul Hojka | APACS |
| Kevin Smith | Independent |
| Gunther Stuhec | SAP AP |
| Jim Wilson | KCX / CIDX |

## 2.1 Acknowledgements

This version OF UN/CEFACT - *XML Naming and Design Rule* was created to foster convergence among Standards Development Organizations (SDOs) with close coordination with these organizations.

- ACORD
- CIDX

- GS1
- HR-XML
- OASIS Universal Business Language (UBL) Technical Committee
- Open Application Group (OAGi)

## 2.2 Disclaimer

The views and specification expressed in this technical specification are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this technical specification.

## 2.3 Contact Information

ATG2 – Jostein Frømyr , EdiSys Consulting AS, Jostein.Fromyr@edisys.no

NDR Project Lead – Mark Crawford, SAL Labs LLC (U.S.), mark.crawford@sap.com

Lead Editor – Michael Rowell, Oracle Corporation, michael.rowell@oracle.com

# 3 Introduction

## 3.1 Summary of Contents of Document

This specification consists of the following Sections and Appendices.

| | |
|---|---|
| Abstract | Informative |
| Table of Contents | Informative |
| Section 1: Status | Informative |
| Section 2: Project Team | Informative |
| Section 3: Introduction | Informative |
| Section 4: Objectives | Normative |
| Section 5: General XML Schema Architecture | Normative |
| Section 6: Application of Context | Informative |
| Section 7: General XML Schema Language Conventions | Normative |
| Section 8: Application of Context in Namespace | Normative |
| Section 9: XML Instance Documents | Normative |
| Section 10: Common Use Cases for Code Lists | Informative |
| Appendix A: Related Documents | Informative |
| Appendix B: Overall Structure | Normative |
| Appendix C: ATG Approved Acronyms and Abbreviations | Normative |
| Appendix D: Business Data Type XML Schema File | Normative |
| Appendix E: Annotation AppInfo Templates | Informative |
| Appendix F: Annotation Documentation Templates | Informative |
| Appendix G: Mapping of CCTS Representation Terms to CCT and BDT | Informative |
| Appendix H: Naming and Design Rules List | Normative |
| Appendix G: Glossary | Normative |

### 3.1.1 Notation

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this specification, are to be interpreted as described in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2119.1. Wherever **`xsd:`** appears in this specification it refers to a construct taken from one of the W3C XML Schema recommendations. Wherever **`ccts:`** appears it refers to a construct taken from the *UN/CEFACT Core Components Technical Specification*.

Example – A representation of a definition or a rule. Examples are informative.

[Note] – Explanatory information. Notes are informative.

[Rn] – Identification of a rule that requires conformance. Rules are normative. In order to ensure continuity across versions of the specification, rule numbers are randomly generated. The number of a rule that is deleted will not be re-issued. Rules that are added will be assigned a previously unused random number.

**`Courier`** – All words appearing in bolded **`courier`** `font` are values, objects or keywords.

When defining rules, the following annotations are used:

**`[ ]`** = optional

**`< >`** = variable

**`|`** = choice

## 3.2 Audience

.The audience for this UN/CEFACT - *XML Naming and Design Rules* Technical Specification are:

- Members of the UN/CEFACT Applied Technologies Group who are responsible for development and maintenance of UN/CEFACT XML Schema
- The wider membership of the other UN/CEFACT Groups who participate in the process of creating and maintaining UN/CEFACT XML Schema definitions
- Designers of tools who need to specify the conversion of user input into XML Schema definitions adhering to the rules defined in this document.
- Designers of XML Schema definitions outside of the UN/CEFACT Forum community. These include designers from other standards organizations and companies that have found these rules suitable for their own organizations.

---

Key words for use in RFCs to Indicate Requirement Levels - Internet Engineering Task Force, Request For Comments 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt?number=2119

# 4 Objectives

## 4.1 Goals of the Technical Specification

This technical specification has been developed to provide for XML standards based expressions of semantic data models representing business information exchanges. It can be employed wherever business information is being shared in an open environment using XML Schema to define the structure of business content. It describes and specifies the rules and guidelines UN/CEFACT will use for developing XML schema and schema documents based on CCTS conformant artefacts and information models developed in accordance with the UN/CEFACT CCTS Technical Specification Version 3.0.

## 4.2 Requirements

Users of this specification should have an understanding of basic data modelling concepts, basic business information exchange concepts and basic XML concepts.

### 4.2.1 Conformance

Designers of XML schema in governments, private sector, and other standards organizations external to the UN/CEFACT community have found this specification suitable for adoption. To maximize reuse and interoperability across this wide user community, the rules in this specification have been categorized to allow these other organizations to create conformant XML schema while allowing for discretion or extensibility in areas that have minimal impact on overall interoperability.

Accordingly, applications will be considered to be in full conformance with this technical specification if they comply with the content of normative sections, rules and definitions.

<table>
<tr>
<td>[R B998]</td>
<td>Conformance SHALL be determined through adherence to the content of the normative sections and rules. Furthermore each rule is categorized to indicate the intended audience for the rule by the following:
<table>
<tr><th colspan="2">Rule Categorization</th></tr>
<tr><th>ID</th><th>Description</th></tr>
<tr><td>1</td><td>Rules which must not be violated by individual organizations else conformance and interoperability is lost – such as named types.</td></tr>
<tr><td>2</td><td>Rules which may be tailored for individual organizations while still conformant to the NDR structure – such as namespace string contents and namespace tokens.</td></tr>
<tr><td>3</td><td>Rules which may be modified by individual organizations while</td></tr>
</table>
</td>
<td>1</td>
</tr>
</table>

| | | |
|---|---|---|
| | still conformant to agreed upon data models – such as the use of global or local element declarations. |
| 4 | Rules that if violated loose conformance with the CEFACT data/process model – such as **`xsd:redefine`**, **`xsd:any`**, and **`xsd:substitutionGroups`**. |
| 5 | Rules that relate to extension that are not used by UN/CEFACT and have specific restrictions on their use by other than CEFACT organizations. |
| 6 | Rules that relate to extension that are determined by specific organizations. |
| 7 | Rules that can be modified while not changing instance validation capability. |

Category 1, 4 and 5 rules can not be modified. Category 2, 3, 6, and 7 may be tailored within the limits identified in the rule and related normative text.

## 4.3 Caveats and Assumptions

The schema created as a result of employing this specification should be made publicly available in a universally freely accessible library as schema documents. UN/CEFACT will maintain their XML schema as published documents in an ebXML compliant registry and make its contents available to any government, individual or organization who wishes access.

Although this specification defines schema components as expressions of core component artefacts, it can also be used by non-CCTS developers for other class based expressions of logical data models and information exchanges.

This specification does not address transformations via scripts or any other means. It does not address any other representation of Core Component artefacts. For example, OWL, Relax NG, XMI and others are outside the scope of this document.

### 4.3.1 Guiding Principles

The following guiding principles were used as the basis for all design rules contained in this specification.

- Relationship to UMM – UN/CEFACT XML Schema defintion will be based on UMM metamodel adherent Business Process Models.
- Relationship to Information Models – UN/CEFACT XML Schema will be based on information models developed in accordance with the UN/CEFACT – *Core Components Technical Specification*.
- XML Schema Creation – UN/CEFACT XML Schema design rules will support XML Schema creation through handcrafting as well as automatic generation.

- ebXML Use – UN/CEFACT XML Schema and XML instance documents shall be easily usable within the ebXML framework and compatible with other frameworks to the maximum extent practicable.
- Interchange and Application Use – UN/CEFACT XML Schema and XML instance documents are intended for business-to-business and application-to-application use.
- Tool Use and Support - The design of UN/CEFACT XML Schema will not make any assumptions about sophisticated tools for creation, management, storage, or presentation being available.
- Legibility - UN/CEFACT XML instance documents should be intuitive and reasonably clear in the context for which they are designed.
- Schema Features - The design of UN/CEFACT XML Schema should use the most commonly supported features of W3C XML Schema Recommendation.
- Technical Specifications – UN/CEFACT XML Naming and Design Rules will be based on Technical Specifications holding the equivalent of W3C recommended status.
- XML Schema Specification – UN/CEFACT XML Naming and Design rules will be fully conformant with W3C XML Schema Recommendation.
- Interoperability - The number of ways to express the same information in a UN/CEFACT XML Schema and UN/CEFACT XML instance document is to be kept as close to one as possible.
- Maintenance – The design of UN/CEFACT XML Schema must facilitate maintenance.
- Context Sensitivity - The design of UN/CEFACT XML Schema must ensure that context-sensitive document types are not precluded.
- Relationship to Other Namespaces - UN/CEFACT XML design rules will be cautious about making dependencies on other namespaces.
- Legacy formats - UN/CEFACT XML Naming and Design Rules are not responsible for sustaining legacy formats.

# 5 XML Schema Architecture

This section defines rules related to general XML Schema constructs these include:

- Overall XML Schema Structure
- Relationship to CCTS
- Naming and Modeling Constraints
- Reusability Scheme
- Message Assembly Considerations
- Namespace Scheme
- XML Schema Files
- Schema Location
- Versioning

## 5.1 Overall XML Schema Structure

UN/CEFACT has determined that the World Wide Web Consortium (W3C) XML Schema Recommendation is the schema definition language with the broadest adoption. Accordingly, all UN/CEFACT XML Schema definitions will be expressed in XML Schema. All references to XML Schema will be as XML Schema. References to XML Schema defined by UNCEFACT will be as UN/CEFACT XML Schema.

| | | |
|---|---|---|
| [R 8059] | All XML Schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures Second Edition and XML Schema 1.1 Part 2: Datatypes. | 1 |

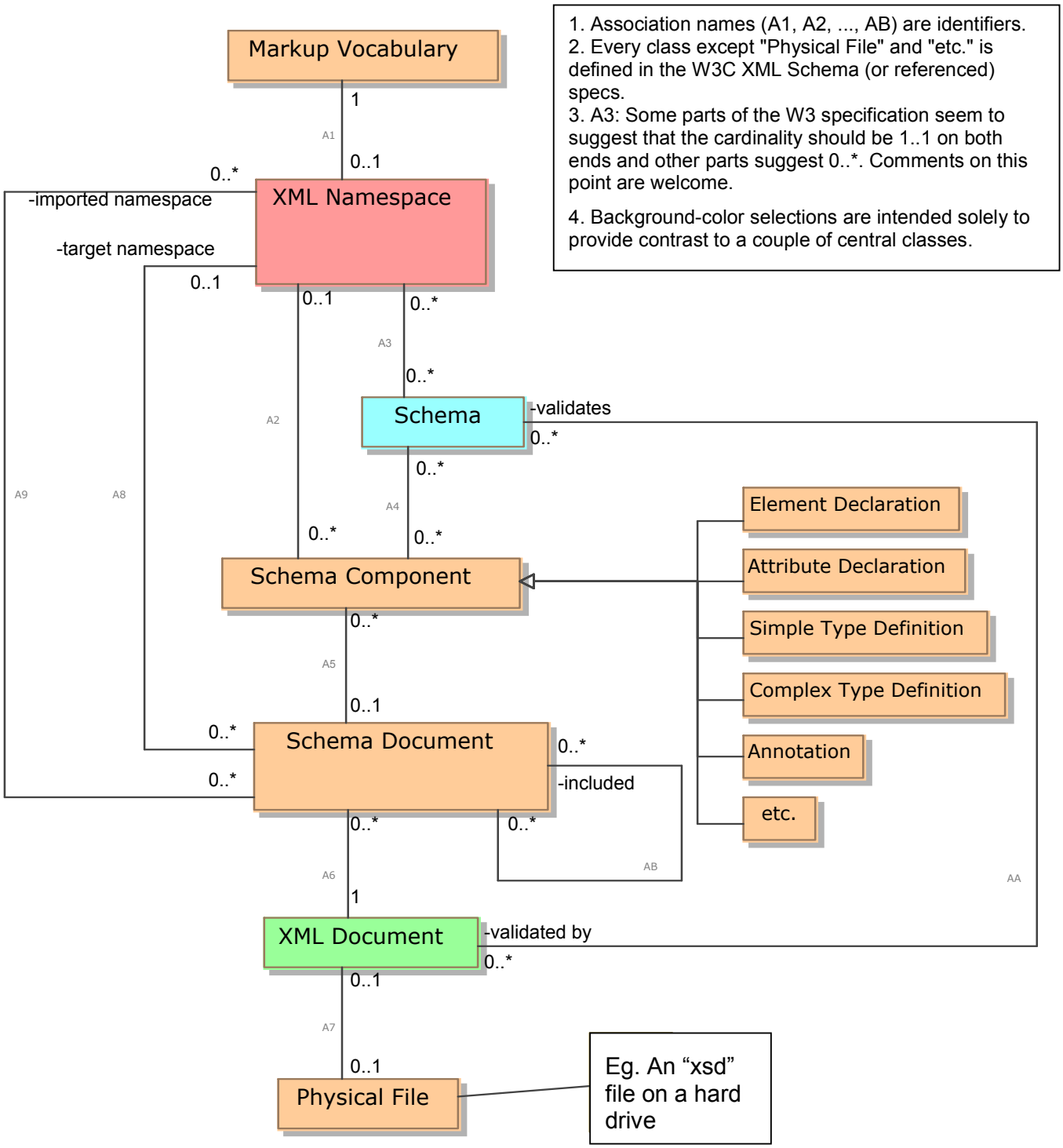
The W3C is the recognized source for XML specifications. W3C specifications can hold various status. Only those W3C specifications holding recommendation status are considered by the W3C to be stable specifications.

| | | |
|---|---|---|
| [R 935C] | All conformant XML instance documents MUST be based on the W3C suite of technical specifications holding recommendation status. | 1 |

To maintain consistency in lexical form, all UN/CEFACT XML Schema need to use a standard structure for all content. This standard structure is contained in Appendix B.

| | | |
|---|---|---|
| [R 9224] | XML Schema MUST follow the standard structure defined in Appendix B of this document. | 1 |

The W3C XML Schema specification uses specific terms in defining the various aspects of a W3C XML Schema. These terms and concepts are used without change in this NDR specification. Figure 5-1, shows these.




340 **Figure 5-1 W3C XML Schema terms and concepts.**

## 341 5.2 Relationship to CCTS

342 All UN/CEFACT business information modelling and business process modelling
343 employ the methodology and model described in UN/CEFACT CCTS.

### 5.2.1 CCTS

CCTS provides a way to identify, capture and maximize the re-use of business information to support and enhance information inter-operability.

The foundational concepts of CCTS are Core Components (CC) and Business Information Entities (BIE). Core Components are building blocks that can be used for all aspects of data, information modelling and information exchange. Core Components are used for creating interoperable business process models and business documents.

Core components are conceptual models that are used to define Business Information Entities (BIEs). The BIEs are the logical data model object used for information exchanges. BIEs are created through the application of context that may:

- Be qualified to provide a unique business semantic,
- Specify a restriction from the underlying CC.

Core Components encompass Aggregate Core Components (ACCs) and Basic Core Components (BCCs), and Association Core Components (ASCCs). Business Information Entities (BIE) encompasses Aggregate Business Information Entities (ABIEs), Basic Business Information Entities (BBIEs), and Association Business Information Entities (ASBIEs).

The CCTS model for BIEs includes

- Common information that are expressed in the annotation documentation in the XML Schema
- Localized information that while expressed in the model is not expressed in the XML Schema.
- Usage Rules that are expressed in the annotation application information in the XML Schema.

### 5.2.2 The XML Schema Components

UN/CEFACT XML Schema design rules are closely coupled with CCTS. UN/CEFACT XML Schema will be developed from fully conformant Business Information Entities that are based on fully conformant Core Components. Figure 5-2 shows the relationship between CCTS Core Components (CCs) artefacts, Business Information Entities (BIEs) artefacts and XML Schema Components. XML Schema Components as defined in Figure 5-2.

Note:

CCTS specifies DataTypes, CCs and BIEs. The columns in Figure 5-2 indicate the conceptual CC Model view and the entity BIE Model view and the how these are translated to XML Schema.

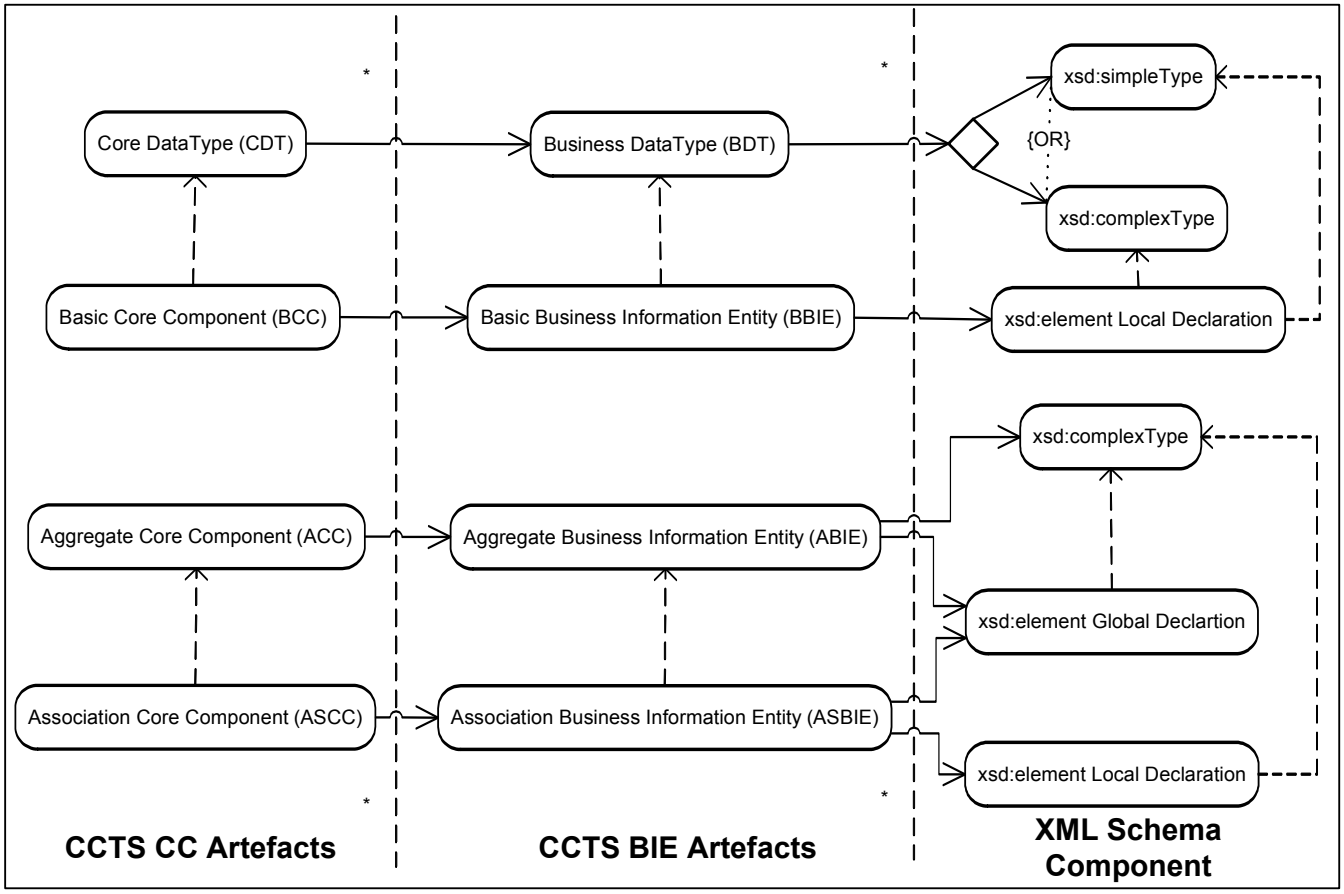



**Figure 5-2 Transitions between CCTS Artefacts and XML Schema Components**

The boxes in the CCTS columns reflect CCTS artifacts:

- Core Components (CC)
  - Core Data Types (CDT)
  - Basic Core Components (BCC)
  - Aggregate Core Components (ACC)
  - Association Core Components (ASCC)
- Business Information Entities (BIE)
  - Business Data Types (BDT)
  - Basic Business Information Entities (BBIE)
  - Aggregate Business Information Entities (ABIE)
  - Association Business Information Entities (ASBIE)

The solid arrows flowing from the CC to the BIE column show the direct mapping of the artefacts from CC to BIEs as defined by CCTS.

The boxes in the XML Schema Components column reflect the XML Schema Components used to express the given BIE. The boxes in the XML Schema Components column show the XML Schema Components being used:

- XML Schema Components

- simpleType
- complexType
- Local element declaration
- Global element declaration

The solid arrow flowing between the BIE column and the XML Schema Component column show the direct mapping from the BIE to the XML Schema Component used to represent it. The dotted arrows with the XML Schema Component column indicate that the given element makes use of type artefact pointed to by the arrow.

Specific ABIEs are identified as the business information payload (Message Assembly). These business information payload (Message Assembly) like all ABIEs are represented as a type definition (**`xsd:complexType`**) and global element (**`xsd:element`**) declaration in an UN/CEFACT XML Schema. The difference in this case is that the Message Assembly recognizes this global element declaration and the type (**`xsd:complexType`**) represents the document level ABIE. The global element is designated as the root element of the UN/CEFACT conformant XML Instances.

Whether an ASBIE uses a local or global element depends upon the type of association (**`AggregationKind=shared`** or **`AggregationKind=composition`**) specified in the model. An ASBIE will be declared as either a local element or as a global element.

- If the ASBIE is a “composition” association (AggregationKind is composition). The ASBIE is declared as a local element (**`xsd:element`**) within the type (**`xsd:complexType`**) representing the associating ABIE. This local element (**`xsd:element`**) makes use of the type (**`xsd:complexType`**) of associated ABIE.
- If it is a “shared” association (AggregationKind is shared). The ASBIE is referenced as a global element (**`xsd:element`**) within the type representing the associating ABIE. The global element (**`xsd:element`**) is declared in the same namespace as the associating ABIE and makes use of the type (**`xsd:complexType`**) of the associated ABIE.

A BBIE is declared as a local element within the **`xsd:complexType`** representing the parent ABIE. The BBIE is based on a (is of type) Business Data Type (BDT).

A BDT is defined as either a **`xsd:complexType`** or **`xsd:simpleType`**. From a modeling perspective BDT’s are based on Core Data Types (CDT). This relationship is not represented in the corresponding XML Schemas. XML Schema built-in data types are to be used whenever the facets of the built-in data type are equivalent to the CCT supplementary components for that data type.

### 5.2.3 Context Categories

The CCTS identifies a set of context categories that affect the resulting context specific BIEs that are created from the CCs. This NDR specification captures all of these context categories through the use annotation application information (**`<xsd:annotation> <xsd:appInfo>`**) element accompanying each element

declaration in UN/CEFACT Schemas. The structure of this information is provided later in this technical specification.

Additionally, each organization adhering to this specification will choose a context category value to incorporate into their namespace. This context category should be the dominant context category for their use. For all UN/CEFACT XML Schema the context category expressed in the namespace is the Business Process.

## 5.3 Naming and Modelling Constraints

UN/CEFACT XML Schemas are derived from components created through the application of CCTS and UN/CEFACT Modelling Methodology (UMM) process modelling and data analysis. UN/CEFACT XML Schema contain XML Schema Components that follow the naming and design rules in this specification. These naming and design rules have taken advantage of the features of the XML Schema specification. In many cases this results in the truncation of the CCTS Dictionary Entry Names (DENs). However, the fully conformant CCTS DENs of the underlying CCTS artefacts are preserved as part of the annotation documentation (**<xsd:annotation> <xsd:documentation>**) element accompanying each element declaration in UN/CEFACT XML Schemas. The CCTS DEN can be reconstructed by using XPath expressions. The fully qualified XPath (FQXP) ties the information to its standardized semantics as described in the underlying CCTS construct and CCTS DEN, while the XML element or attribute name is a truncation that reflects the hierarchy inherent in the XML construct.

The FQXP anchors the use of a construct to a particular location in a business information payload. The dictionary definition identifies any semantic dependencies that the FQXP has on other elements and attributes within the UN/CEFACT library that are not otherwise enforced or made explicit in its structural definition. The dictionary serves as a traditional data dictionary, and also some of the functions of a traditional implementation guide.

| | | |
|---|---|---|
| [R A9E2] | Each element or attribute XML name MUST have one and only one fully qualified XPath (FQXP). | 1 |

All rules on element naming are constructed so that a part of the fully qualified XPath will always represent the CCTS dictionary entry name of the corresponding ABIE, BBIE, ASBIE or BDT.

Example 5-1 shows a FQXP for Address Coordinate LatitudeMeasure and Organization Location Name.

**Example 5-1: Fully Qualified XPath**

```
Address/Coordinate/LatitudeMeasure
Organisation/Location/Name
```

The official language for UN/CEFACT is English. All official XML constructs as published by UN/CEFACT will be in English. XML and XML Schema development work may very well occur in other languages, however official submissions for inclusion in the UN/CEFACT XML Schema library must be in English. Other

language translations of UN/CEFACT published XML and XML Schema Components are at the discretion of the users.

| | | |
|---|---|---|
| [R AA92] | Element, attribute and type names MUST be composed of words in the English language, using the primary English spellings provided in the Oxford English Dictionary. | 1 |

Following commonly used best practice, LowerCamelCase (LCC) is used for naming attributes and UpperCamelCase (UCC) is used for naming elements and types. LowerCamelCase capitalizes the first character of each word except the first word and compounds the name. UpperCamelCase capitalizes the first character of each word and compounds the name.

Examples 5-2 through 5-6 show examples of what is allowed and not allowed.

**Example 5-2: Attribute**

Allowed

```
<xsd:attribute name="unitCode" .../>
```

**Example 5-3: Element**

Allowed

```
<xsd:element name="LanguageCode" ...>
```

**Example 5-4: Type**

Allowed

```
<xsd:complexType name="DespatchAdviceCodeType">
```

**Example 5-5: Singular and Plural Concept Form**

Allowed - Singular:

```
<xsd:element name="GoodsQuantity" ...>
```

Not Allowed - Plural:

```
<xsd:element name="ItemsQuantity" ...>
```

**Example 5-6: Non-Letter Characters**

Not Allowed

```
<xsd:element name="LanguageCode8" ...>
```

| | | |
|---|---|---|
| [R 9956] | LowerCamelCase (LCC) MUST be used for naming attributes. | 1 |

| | | |
|---|---|---|
| [R A781] | UpperCamelCase (UCC) MUST be used for naming elements and types. | 1 |
| [R 8D9F] | Element, attribute and type names MUST be in singular form unless the concept itself is plural. | 1 |
| [R BFB0] | Element, attribute and type names MUST be drawn from the following character set: a-z and A-Z. | 1 |

While CCTS allows for the use of periods, spaces and other separators in the dictionary entry name. XML best practice is to not include these in an XML tag name. Additionally, XML 1.0 specifically prohibits the use of certain reserved characters in XML tag names.

Examples 5-7 and 5-8 show examples of what is allowed and not allowed.

**Example 5-7: Spaces in Name**

Not Allowed

```
<xsd:element name="Customized_ Language. Code:8" ...>
```

**Example 5-8: Acronyms and Abbreviations**

Allowed – ID is an approved abbreviation

```
<xsd:attribute name="currencyID"
```

Not Allowed – Cd is not an approved abbreviation, if it was an approved abbreviation it must appear in all upper case

```
<xsd:simpleType name="temperatureMeasureUnitCdType>
```

| | | |
|---|---|---|
| [R AB19] | XML element, attribute and type names constructed from dictionary entry names MUST NOT include periods, spaces, or other separators; or characters not allowed by W3C XML 1.0 for XML names. | 1 |
| [R 9009] | XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other word truncations, except those included in the defining organizations list of approved acronyms and abbreviations. | 1 |
| [R BFA9] | The acronyms and abbreviations listed by the defining organization MUST always be used in place of the word or phrase they represent. | 1 |
| [R 9100] | Acronyms MUST appear in all upper case except for when the acronym is the first set of characters of an attribute in which case they will be all lower case. | 1 |

## 5.4 Reusability Scheme

UN/CEFACT is committed to an object based approach for its process models and core component implementation efforts as supported in both UMM and CCTS.

UN/CEFACT considered adopting a type based approach (named types), a type and element based approach, or an element based approach. A type based approach for XML management provides the closest alignment with the process modelling methodology described in UMM. Type information is beginning to be accessible when processing XML instance documents. Post schema-validation infoset (PSVI) capabilities are beginning to emerge that support this approach, such as “data-binding” software that compiles schema into ready-to-use object classes and is capable of manipulating XML data based on their types. The most significant drawback to a type based approach is the risk of developing an inconsistent element vocabulary where elements are declared locally and allowed to be reused without regard to semantic clarity and consistency across types. UN/CEFACT manages this risk by carefully controlling the creation of BBIEs and ASBIEs with fully defined semantic clarity that are only usable within the ABIE in which they appear. This is accomplished through the relationship between BBIEs, ASBIEs and their parent ABIE and the strict controls put in place for harmonization and approval of the semantic constructs prior to their XML Schema instantiation.

A purely type based approach does, however, limit the ability to reuse elements, especially in technologies such as Web Services Description Language (WSDL).

For these reasons, UN/CEFACT implements a “hybrid approach” that provides benefits over a pure type based approach. Most significantly it increases reusability of library content both at the modelling and XML Schema level.

The key principles of the “hybrid approach” are:

- All classes (Invoice, Seller_Party, Buyer_Party, Invoice_Trade.Line.Item and Billed_Delivery in Figure 5-3) are declared as a **`xsd:complexType`**.
- All attributes of a class are declared as a local **`xsd:element`** within the corresponding **`xsd:complexType`**.
- UML Aggregation Kind composition associations (e.g. Invoice_Trade.Line.Item and Billed_Delivery in Figure 5-3) will result in a locally declared **`xsd:element`** with a globally declared **`xsd:complexType`**. A composition **`ASBIE is defined`** as a specialized type of ASBIE that represents a composition relationship between the associating ABIE and the associated ABIE.
- An association that is not defined as composition (e.g. Invoice.Buyer. Buyer_Party, Invoice. Seller. SellerParty in Figure 5-3) will result in a globally declared **`xsd:element`** with a globally declared **`xsd:complexType`**. In specific cases the schema will also allow the global element to be referenced via the key/keyRef referencing mechanism.

The rules pertaining to the ‘hybrid approach’ are contained in sections *8.2.3 Type Definitions* and *8.2.4 Type Definitions* for type and element declaration.

Figure 5-3 shows an example UML model and Example 5-9 shows the resulting XML Schema declaration that results from the translation from UML to XML Schema

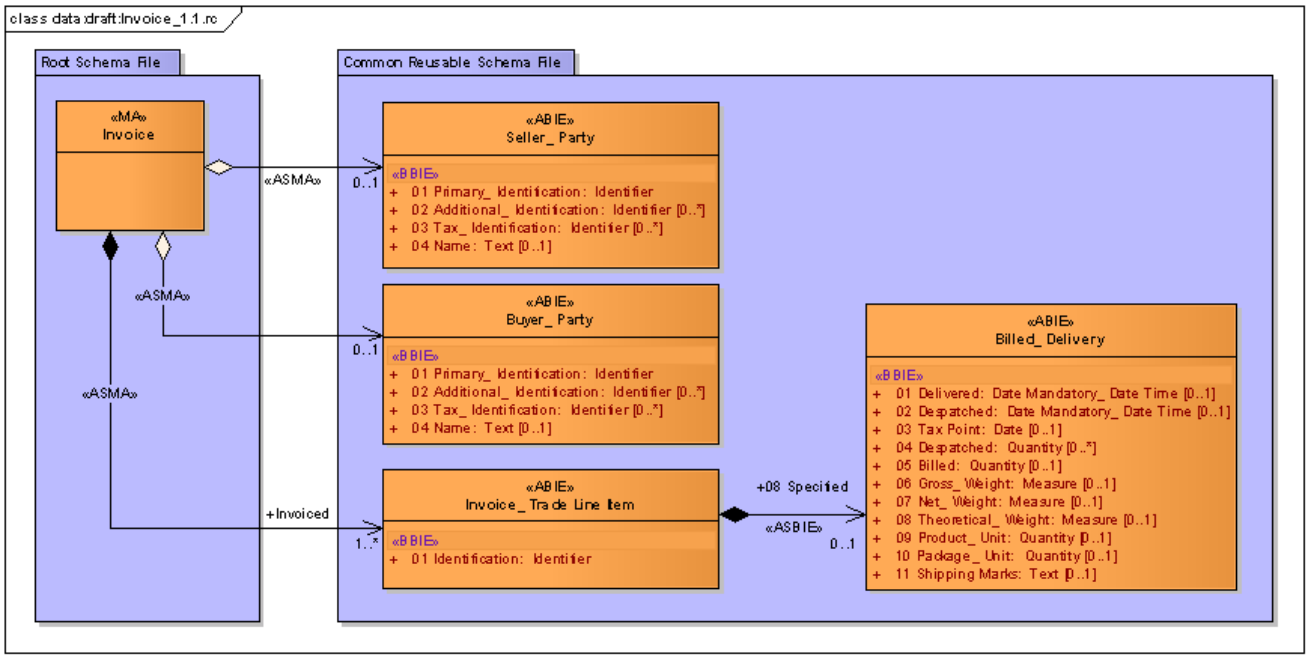



**Figure 5-3 UML Model Example**

**Example 5-9: XML Schema declarations representing Figure 5-3.**

```
  <xsd:element name="InvoiceRequest" type="rsm:InvoiceType"/>

  <xsd:element name="BuyerParty" type="ram:BuyerPartyType"/>
   <xsd:element name="InvoiceTradeLineItem" type="ram:InvoiceTradeLineItemType"/>
   <xsd:element name="SellerParty" type="ram:SellerPartyType"/>

 <xsd:complexType name="InvoiceType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element ref="ram:SellerParty"/>
                <xsd:element ref="ram:BuyerParty"/>
                <xsd:element name="InvoiceTradeLineItem"
type="ram:InvoiceTradeLineItemType" maxOccurs="unbounded"/>
         </xsd:sequence>
  </xsd:complexType>

 <xsd:complexType name="BuyerPartyType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element name="Name" type="bdt:NameType"/>
         </xsd:sequence>
  </xsd:complexType>

 <xsd:complexType name="InvoiceTradeLineItemType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element name="BilledDelivery" type="ram:BilledDeliveryType"/>
         </xsd:sequence>
  </xsd:complexType>

 <xsd:complexType name="BilledDeliveryType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element name="Name" type="bdt:NameType"/>
         </xsd:sequence>
  </xsd:complexType>

 <xsd:complexType name="SellerPartyType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element name="GivenName" type="bdt:NameType"/>
                <xsd:element name="Surname" type="bdt:NameType"/>
```

```
        </xsd:sequence>
  </xsd:complexType>
```

## 5.5 Message Assembly Considerations

### 5.5.1 Implementation of Aggregations – Nesting or Referencing

Since aggregations relate ABIEs that have independent life cycles, the same instance of a particular ABIE may be referenced more than once within a message. An example, in the Insurance Industry, a ClaimNotify message shown below in Example 5-10 and Example 5-11 the same Person '**John Smith**' can play the role of "Insured" in the Policy ABIE and the role of "Claimant" in the Claim ABIE. In order to address this, it is possible to use XML referencing mechanism to relate one Person instance to the Policy and Claim instances as an alternate method to nesting information about Person within Policy and Claim.

**Example 5-10: XML Instance using nesting**

```
<ClaimNotify>
......
<Claim>
    <ClaimantParty>
<Name>John Smith</Name>
</ClaimantParty>
<Claim>
......
<Policy>
  <InsuredParty>
        <Name>John Smith</Name>
  </InsuredParty>
</Policy>
<ClaimNotify>
```

**Example 5-11: XML Instance using referencing**

```
<ClaimNotify>
......
<Party key="P1">
<Name>John Smith</Name>
</Party>
<Claim>
<ClaimantParty partyReference="P1"/>
<Claim>
......
<Policy>
  <InsuredParty partyReference="P1"/>
</Policy>
…….
<ClaimNotify>
```

In general, when the level of reuse of an instance ABIE in a message is significant it becomes adequate to use XML referencing for the purpose of removing redundancy from the message and increasing information integrity.

### 5.5.2 Other Usages of XML Referencing

This document also addresses *dynamic referencing* which is described as: Any element composing a message is potentially the target of a reference for the purpose of building dynamic relationships between elements within the message. An important use case is identification of faulty elements for error reporting.

General usage of dynamic referencing requires adding an optional identifier property to XML elements. Such identifiers are typically used to build short XPath expressions pointing to the XML element. Therefore this specification generalizes the addition of an optional identifier attribute to each element defined as **`xsd:complexType`**, as detailed in section *8.2.3 Type Definitions*. Such an attribute can be used for dynamic referencing as well as structural referencing in support of aggregations of ABIEs.

### 5.5.3 Schema Validation Requirements for XML References

#### 5.5.3.1 Structural References between Aggregated ABIEs

For structural references between ABIEs, the level of validation performed by the XML Schema definition of a message should be as strong as if the referenced element would have been defined as a nested child of the element that references it. Thus, the schema must strictly enforce identity constraints, i.e.:

1. Check uniqueness of the identifiers of the referenced elements
2. Check that the references match the identifiers of the corresponding referenced elements.

This specification mandates **`key/keyRef`** as the XML referencing technique to be used, instead of **`Id/IdRef`**, as detailed in section *8.1.5 Declaration of the Referencing Constraints*.

Referencing between ABIEs occur in the boundaries of a particular 'scoping element' in the XML document tree (scoping element means an element in the hierarchy of the XML document under which a closed set of references can be defined). Most often the scoping element will be the message root element but it can also be another element lower in the hierarchy. The XML Schema language requires that the key-keyref constraints be defined within a scoping element.

#### 5.5.3.2 Dynamic References

For dynamic references schema validation is not required. Since dynamic referencing is only used for ancillary purposes, it is not deemed essential to enforce uniqueness of identifiers in the schema when they are not involved in structural referencing. Uniqueness of such identifiers should be granted by use of adequate algorithms for the generation of the identifiers. This will avoid unnecessary complexity of the identity constraints.

| | | |
|---|---|---|
| [R B8B6] | Empty elements MUST NOT be used, except when their definition include an identifier attribute that serves to reference another element via schema identity constraints. | 1 |

### 5.5.4 Message Assembly Definition Requirements

Figure 5-4 shows the Message Assembly Metamodel. The following is assumed for generating a message schema:

- The message structure is specified by a model as defined by the UN/CEFACT Business Message Template document, in the form of a single Message Assembly (MA) component consisting of a hierarchy of Association Message

Assemblies (ASMAs), including ASMAs that may have been derived from other ASMAs within the same MA.

- Each ASMA recursively contains an ordered list of child ASMAs/ASMBIEs down to the bottom of the hierarchy.
- Should referencing between specific MBIEs be required for the message in the scope of a higher level MA or MBIE. A higher level MA or MBIE must define the list of MBIEs that are implemented as referenced rather than nested properties. This will allow the identity constraints to be generated in the message schema.

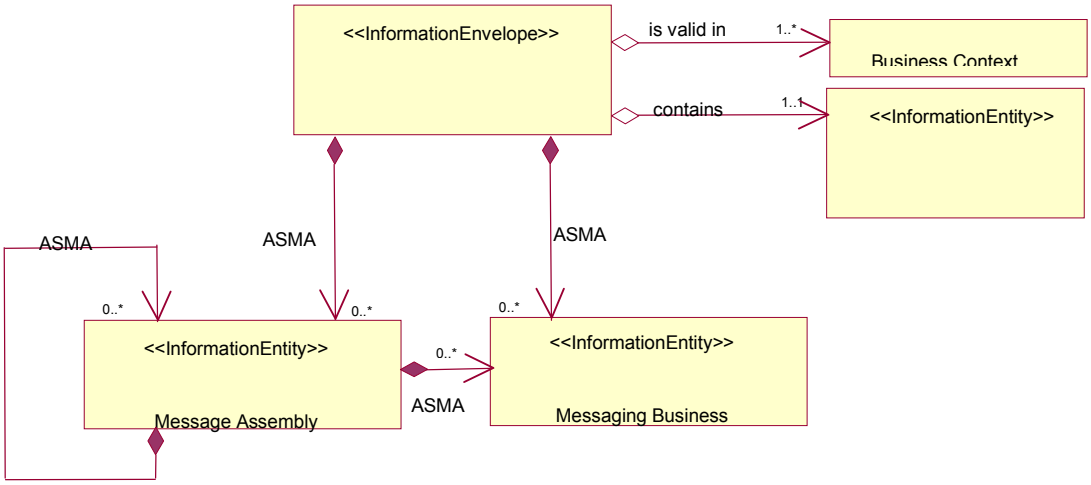



**Figure 5.4: Business Message Template Metamodel**

## 5.6 Namespace Scheme

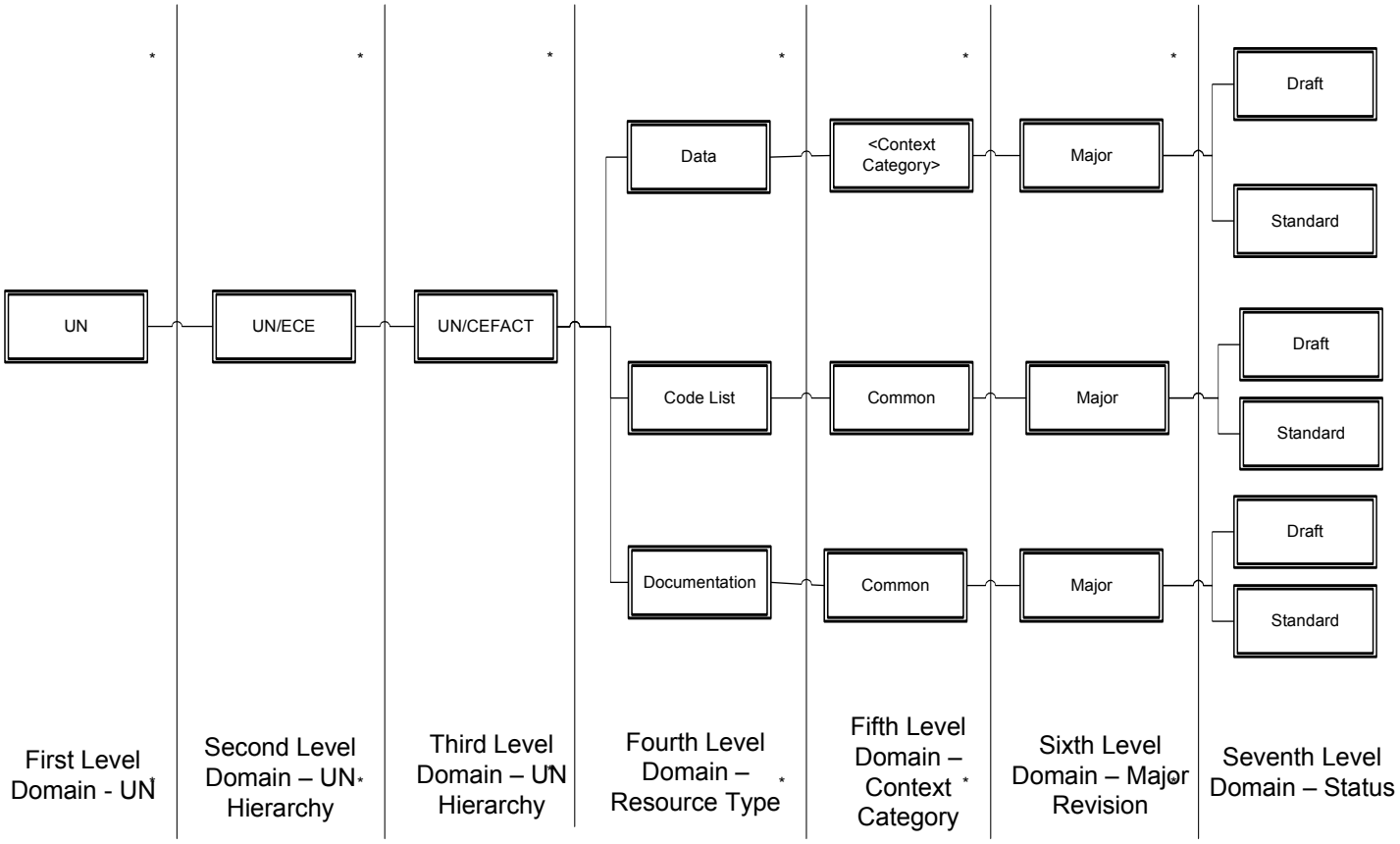
A namespace is a collection of elements, attributes and types that serve to uniquely distinguish the collection in a given business context.

"A XML namespace is identified by a URI reference [RFC3986]; element and attribute names may be placed in an XML namespace…".[2] UNCEFACT assigns XML artifacts to a UNCEFACT namespace. These namespaces reflect logical groupings as shown in Figure 5-5.

Each organization that intends to adhere to this specification will assign their XML Schema defined content in a namespace that reflects the name of the organization and context category in which the XML Schema is defined.

| | | |
|---|---|---|
| [R 984C] | Each organization's XML Schema components MUST be assigned to a namespace for that organization. | 1 |

[2] http://www.w3.org/TR/2006/REC-xml-names-20060816/




714 **Figure 5-5: UN/CEFACT Namespace Scheme**

715 ### 5.6.1 Namespace Uniform Resource Identifiers

716 Namespaces must be persistent. Namespaces should be resolvable. A URI is used
717 for identifying a namespace. Within the URI space, options include Uniform
718 Resource Locators (URLs) and Uniform Resource Names (URNs). A URN has an
719 advantage in that it is persistent. A URL has an advantage in that it is most often
720 resolvable.

721 To ensure consistency, each namespace identifier will have the same general
722 structure. The URN namespace structure will follow the provisions of *Internet*
723 *Engineering Task Force (IETF) Request For Comments (RFC) 2141 – URN Syntax*.

724 The URN format will be:

725 **urn:<organization>:<org hierarchy>[:<org hierarchy**
726 **level>]*:<schematype>:<context category>:<major>:<status>**

727 The URL namespace structure will follow the provisions of Internet Engineering Task
728 Force (IETF) Request For Comments (RFC) 1738 – Uniform Resource Locators
729 (URL)

730 The URL format will be:

731 **http://<organization>/<org hierarchy>[/<org hierarchy**
732 **level>]*/<schematype>/<context category>/<major> /<status>**

Where:

- organization – An identifier of the organization providing the standard.
- org hierarchy – The first level of the hierarchy within the organization providing the standard.
- org hierarchy level – Zero to n level hierarchy of the organization providing the standard.
- schematype – A token identifying the type of schema module: data|codelist|documentation
- context category – The context category [business process] for UN/CEFACT from the UN/CEFACT catalogue of common business processes. Other values may be used by the other organizations.
- major – The major version number
- status – The status of the schema as: **draft|standard**.

<table>
<tr>
<td>[R 8E2D]</td>
<td>The XML Schema namespaces MUST use the following pattern:
<table>
<tr><td>URN:</td><td>urn:<organization>:<org hierarchy>[:<org hierarchy level>]*:<schematype>:<context category>:<major>:<status></td></tr>
<tr><td>URL:</td><td>http://<organization>/<org hierarchy>[/<org hierarchy level>]*/<schematype>/context category/<major>/<status></td></tr>
</table>
Where:
<ul>
<li>organization – An identifier of the organization providing the standard.</li>
<li>org hierarchy – The first level of the hierarchy within the organization providing the standard.</li>
<li>org hierarchy level – Zero to n level hierarchy of the organization providing the standard.</li>
<li>schematype – A token identifying the type of schema module: data|codelist|documentation</li>
<li>context category – The context category [business process] for UN/CEFACT from the UN/CEFACT catalogue of common business processes. Other values may be used by the other organizations.</li>
<li>major – The major version number</li>
<li>status – The status of the schema as: draft|standard.</li>
</ul>
</td>
<td>3</td>
</tr>
</table>

UN/CEFACT has determined that URNs are most appropriate as persistence is of a higher priority for UN/CEFACT. Furthermore, UN/CEFACT recommends that URNs

be used by other organizations that use this NDR. However, each organization using this NDR ultimately must decide for themselves which is more important to them.

| | | |
|---|---|---|
| [R 8CED] | UN/CEFACT namespaces MUST be defined as Uniform Resource Names. | 3 |

**Example 5-12: Namespace Name at Draft Status**

```
"urn:un:unece:uncefact:data:ordermanagement:1:draft"
```

**Example 5-13: Namespace Name at Specification Status**

```
"urn:un:unece:uncefact:data:odermanagement:1:standard”
```

Once a namespace's content is published, any change that breaks backwards compatibility will require a new namespace.

| | | |
|---|---|---|
| [R B56B] | Published namespace content MUST NOT be changed unless such change does not break backward compatibility. | 1 |

### 5.6.2 Namespace Tokens

Namespace URIs are typically aliased by using tokens rather than citing the entire URI as the qualifier in a qualified name of a given XML Schema File.

As identified in the namespace scheme defined in section *5.6.1 Namespace Uniform Resource Identifiers* will be assigned to namespaces based on a context value category. Namespace tokens representing the namespace will be created using three character representations for each unique value within the chosen context category.

XML Schema files that are defined for common CodeList will use a token that is prefixed with ‘clm’.

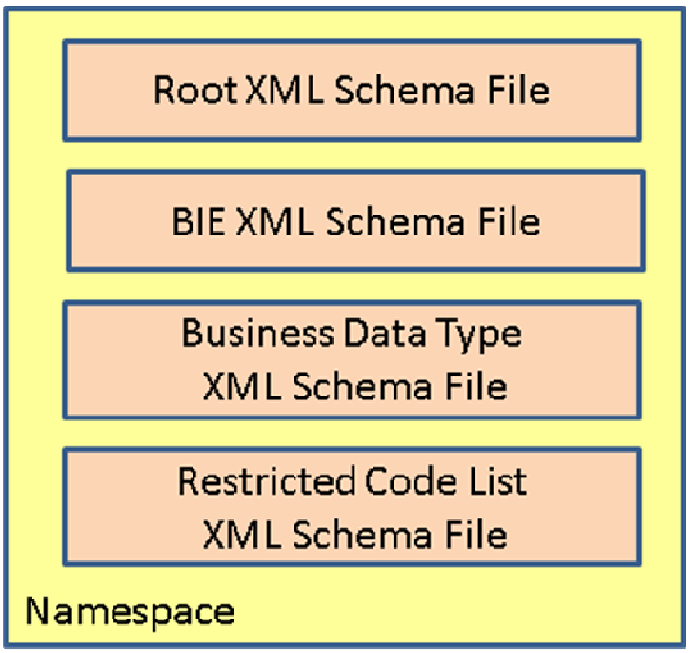
## 5.7 XML Schema Files

A XML Schema file is a schema document realized as a physical file. As defined by the W3C, a schema document represents relevant instantiations of the thirteen defined W3C XML Schema XML Schema Components that collectively comprise an abstract data model.

XML Schema files created from this specification represent abstract data models for messages, CCTS conformant ABIEs, BDTs, restricted code lists and referenced common code lists. Figure 5-6 shows how the messages, CCTS conformant ABIEs, BDTs, and restricted code lists within a given context category are assigned to a single namespace. Since common code lists are applicable to all context categories, each resides in its own namespace.

XML Schema files can be either unique in their functionality, or represent splitting of larger XML Schema files for performance or manageability enhancement. A well thought out approach to the layout provides an efficient and effective mechanism for

780 providing components as needed rather than dealing with complex, multi-focused
781 XML Schema files.




782

783 **Figure 5-6: UN/CEFACT XML Schema Files**

784 UN/CEFACT has defined a number of XML Schema files to support this approach.
785 These XML Schema files are defined for the given context category value.
786 UNCEFACT XML Schema namespaces are defined based upon Business Process
787 Value. For each Business Process Value which is defined as a separate namespace
788 a set of Root XML Schema files (one per business information payload), a common
789 BIE XML Schema File, a BDT XML Schema File, a set of restricted Code List XML
790 Schema File. Furthermore, where common code lists can be used the given Code
791 List XML Schema file may be imported into the BDT XML Schema File and/or
792 Restricted Code List XML Schema file. Dependencies exist among the various files
793 as shown in Figure 5-7.

794 Each of the Root XML Schema files defined with in the given context category
795 namespace (UNCEFCT uses Business Process) always includes the ABIE XML
796 Schema file and the BDT XML Schema file. The ABIE XML Schema file always
797 includes the BDT XML Schema file. The BDT XML Schema file always include zero
798 or more Restricted CodeList XML Schema files, it also always imports zero or more
799 Common Code List XML Schema files.

| | | |
|---|---|---|
| [R 92B8] | The XML Schema file name for files other than code lists MUST be of the form **`<SchemaModuleName>_<Version>.xsd`**, with periods, spaces, or other separators and the words XML Schema File removed. | 3 |
| [R 8D58] | When representing versioning schemes in file names, the period MUST be represented by a lowercase **`p`**. | 3 |

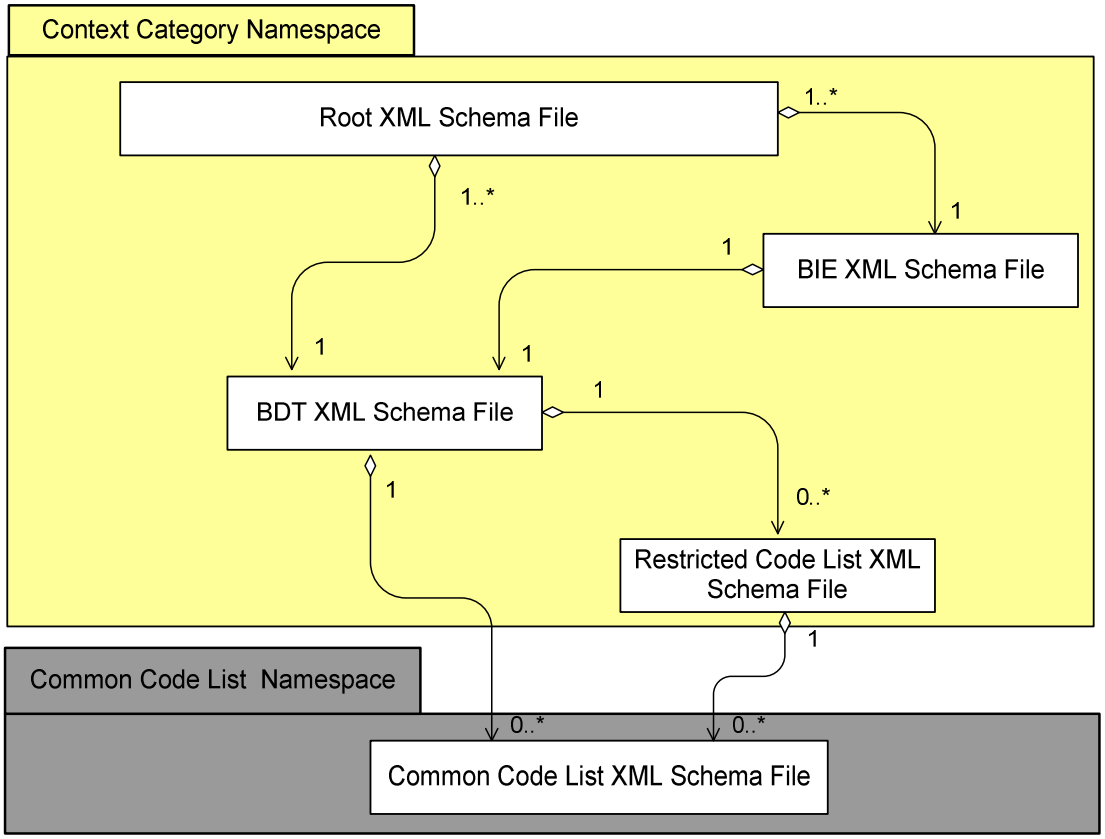



**Figure 5-7: UN/CEFACT XML Schema Modularity Scheme**

### 5.7.1 Root XML Schema Files

This NDR specification requires that the namespace reflect the dominate context category value for the XML Schema files being defined by the defining organization. For UN/CEFACT that dominate context category is the Business Process context category.

The set of root schema files within a given context category are assigned to the namespace of that context category. The UN/CEFACT namespace scheme shown in Figure 5-5 reflects this approach.

Each **`xsd:schema`** element used to define an XML Schema Document will have the namespace declared using **`xsd:targetNamespace`**.

| | | |
|---|---|---|
| [R B387] | Every XML Schema file MUST have a namespace declared, using the **`xsd:targetNamespace`** attribute. | 1 |

The UN/CEFACT modularity approach provides for a reusable BIE XML Schema file and a BDT XML Schema file that are used by a set of Root XML Schema files within the given context category namespace.

The contents of a schema set are so interrelated that proper management dictates that both versioning and namespace of all members of the set be synchronized so that concept collisions are avoided. Schema sets are therefore assigned to a single, versioned namespace.

UN/CEFACT incorporates a XML Schema file modularity scheme that leverages the benefits described in the UN/CEFACT XML Schema artefact repository. There are a number of UN/CEFACT Root XML Schema, each of which expresses a separate business information payload. The Root XML Schema files include the recognized business transactions for the given context category based namespace.

| | | |
|---|---|---|
| [R 9354] | A Root XML Schema file MUST be created for each unique business information payload. | 1 |

To ensure uniqueness, Root XML Schema files will be given unique names that reflect the business function being addressed by the schema. This business function is described in the UN/CEFACT Requirements Specification Mapping (RSM) document as the target business information payload. The business information payload name representing the business function will form the basis for the Root XML Schema file name.

| | | |
|---|---|---|
| [R B3E4] | Each Root XML Schema File MUST be named after the <BusinessInformationPayload> XML Schema File in the documentation within the XML Schema File. | 1 |

This approach enables the use of individual context category focused Root XML Schema files without importing the entire library. Each Root XML Schema will define its own dependencies. A Root XML Schema file should not duplicate common reusable XML constructs contained in the common ABIE XML Schema file for the given context category. Specifically, Root XML Schema files will include other XML Schema files to maximize reuse for the given context category.

| | | |
|---|---|---|
| [R 9961] | A Root XML Schema file MUST NOT replicate reusable constructs available in XML Schema files that can be referenced through **`xsd:include`**. | 1 |

### 5.7.2 Business Data Type XML Schema Files

The CCTS Business Data Types (BDTs) define the value domain for a Basic Business Information Entity. The value domain is defined by selecting from one of the allowed primitives for the BDT and providing additional restrictions if desired through the use of supplementary components. UN/CEFACT publishes a BDT XML Schema File that consists of all BDTs without restriction to the value domain. This schema file resides in its own namespace and is used for reference purposes only. Additional BDT Schema will be created and published as BDT XML Schema Files within the namespace of the context category it supports. Each BDT XML Schema File will have a standardized name that uniquely differentiates it from other UN/CEFACT XML Schema Files.

| | | |
|---|---|---|
| [R AA56] | A Business Data Type XML Schema File MUST be created within | 1 |

| | | |
|---|---|---|
| | each context category based namespace. | |
| [R 847C] | The bdt:BusinessDataType XML Schema File MUST be named 'Business Data Type XML Schema File' in the documentation within the XML Schema File. | 1 |

### 5.7.3 Business Information Entity XML Schema Files

A BIE XML Schema File will be created to contain XML Schema Components used to define the common reusable ABIEs and their BBIEs and ASBIEs within the principal context category used for namespace. Where desired, these BIE XML Schema files may be further compressed for runtime performance considerations if necessary through the creation of a runtime version that only includes those ABIEs necessary to support the root schema including it.

Each BIE XML Schema File will have a standardized name that uniquely differentiates it from other UN/CEFACT XML Schema Files.

| | | |
|---|---|---|
| [R 8238] | One Business Information Entity XML Schema Files MUST be created for the context category that is expressed in the namespace. | 1 |
| [R 8252] | The BusinessInformationEntity XML Schema file MUST be named 'Business Information Entity XML Schema File' by placing the name within the Header documentation section of the file. | 1 |

### 5.7.4 Code List XML Scehema Files

#### 5.7.4.1 Restricted Code List XML Schema Files

A set of restricted code list may be created in cases where a restricted common code list is required or where a code list does not currently exist and one can be identified. These restricted code list are to be defined in the same namespace as the XML Schema that make use of them such that the context category value in which they are valid is present.

Each Code List XML Schema file will contain enumeration values for codes and code values.

Code list schema modules will have a standardized name that uniquely differentiates it from other UN/CEFACT XML Schema Files and external organization generated code list files.

| | | |
|---|---|---|
| [R BD2F] | A Restricted Code Lixt XML Schema File MUST be created for each restricted code list used by a BDT. | 1 |
| [R 942D] | Each Restricted Code List XML Schema File MUST contain enumeration values for both the actual codes and the code values. | 1 |
| [R A62F] | Each Restricted Code List XML Schema File MUST be given a unique name within the namespace it belongs. | 1 |

### 5.7.4.2 Common Code List XML Schema Files

Some common code lists are published by standards organizations to represent a set of commonly accepted codes that are used in a variety of business circumstances and contexts. A reusable Common Code List XML Schema file will be created for each code list that represents a published standard code list.

| | | |
|---|---|---|
| [R 8A68] | Cases where code lists are used within the XML Schema, a Code List XML Schema file MUST be created to convey code list enumerations for each code list being used. | 1 |

Common Code List XML Schema files will have a standardized name that uniquely differentiates it from other UN/CEFACT XML Schema files and other external organization generated code list files.

| | | |
|---|---|---|
| [R B443] | Each Common Code List XML Schema File must be given a unique name that represents the name of the code list and is unique within the namespace it belongs. | 1 |
| [R B0AD] | The name of each **clm:CodeList** XML Schema File as defined in the comment within the XML Schema File MUST be of the form:<br>**<Code List Agency Identifier\|Code List Agency Name><Code List Identification Identifier\|Code List Name>" - Code List XML Schema File"**<br>Where:<br>• Code List Agency Identifier – Identifies the agency that maintains the code list<br>• Code List Agency Name – Agency that maintains the code list<br>• Code List Identification Identifier – Identifies a list of the respective corresponding codes<br>• Code List Name – The name of the code list as assigned by the agency that maintains the code list. | 1 |

**Example 5-14: Name of UN/CEFACT Account Type Code List XML Schema File Name using Identifiers**

```
64437 - Code List XML Schema File
where:
6 = Code list agency identifier for UN/CEFACT as defined in UN/CEFACT code
    list 3055
4437 = Code list identification identifier for Account Type Code in UN/CEFACT
    directory
```

**Example 5-15: Name of UN/CEFACT Security Type Code List XML Schema File Name using Names**

```
Security Initiative Document Security Code - Code List XML Schema File
```

### 5.7.5 Other Standard Bodies BIE XML Schema Files

Other Standards Development Organizations create and make publicly available BIE XML Schema Files. UN/CEFACT will only import these other SDO BIE XML Schema Files when their contents are in strict conformance to the requirements of the CCTS technical specification and this NDR technical specification.

Strict conformance means that a schema is conformant to category 1, 2, 3, 4 and 7 rules as defined in rule B998.

| | | |
|---|---|---|
| [R B564] | Imported XML Schema Files MUST be fully conformant to category 1, 2, 3, 4 and 7 rules as defined in Rule B998. | 4 |
| [R 9733] | Imported XML Schema File components MUST be derived using these NDR rules from artifacts that are fully conformant to the latest version of the UN/CEFACT Core Components Technical Specification. | 4 |

## 5.8 Schema Location

Schema locations:

- Are required to be in the form of a URI scheme;
- Are associated to the namespace of the file being accessed;
- Are typically defined as URLs because of resolvability limitations of URNs;
- Can be defined as absolute path or relative paths.

According to the W3C XML Schema specification, part 0, the schemaLocation attribute "… provides hints from the author to a processor regarding the location of a schema document. The author warrants that these schema documents are relevant to checking the validity of the document content, on a namespace by namespace basis."[3] The value provided in the **`xsi:schemaLocation`** attribute is "…only a hint and some processors and applications will have reasons to not use it." Thus the presence of these hints does not require the processor to obtain or use the cited schema documents, and the processor is free to use other schemas obtained by any suitable means, or to use no schema at all.

In practical implementations many XML tools attempt to acquire resources using the schema location attribute. The implication of the schemaLocation attribute pointing to an absolute path (e.g., hard-drive location; URL) is that when tools attempt to acquire the resources and they are not available at the specified location, the tool may raise errors. In the case of URL-formatted schemaLocation values, this might occur after a seemingly lengthy timeout period, a period in which other work cannot be done. On the other hand, relative paths increase the likelihood that resources will be readily available to tools (assuming well organized schema files). Thus using an absolute path approach with URL-formatted schemaLocation values often represents a challenge in practical implementations as it requires open internet connections at

[3] http://www.w3.org/TR/xmlschema-0/#schemaLocation

run-time (due to tool implementations) and is seen as a security issue by a number of implementers.

Providing the schemaLocation value as a relative path provides an overall improvement in user productivity, including off-line use. It is important to note that this approach doesn't prohibit making resources available on-line (much in the same way that HTML documents frequently provided references to relative locations for images).

| | | |
|---|---|---|
| [R 8F8D] | Each **xsd:schemaLocation** attribute declaration MUST contain a resolvable URL. This may include a relative path reference from the location of the current XML Schema file. | 2 |

**Example 5-16: Example of relative path schemaLocation.**

```
<xsd:import namespace="urn:un:unece:uncefact:ordermanagementdata:draft:1"
schemaLocation="../../data/draft/BusinessDataType_1p0.xsd"/>
```

## 5.9 Versioning Scheme

The UN/CEFACT versioning scheme consists of:

- Status of the XML Schema file,
- A major version number,
- A minor version number and
- A revision number.

These values are declared in the version attribute in the **xsd:schema** element. The major version number is also reflected in the namespace declaration for each XML Schema file (R 8E2D).

| | | |
|---|---|---|
| [R BF17] | The **xsd:schema** version attribute MUST always be declared. | 1 |
| [R 84BE] | The **xsd:schema** version attribute MUST use the following template:<br>**<xsd:schema ... version="Draft" \| "Standard" _<major>"p"<minor>["p"<revision>]">**<br>Where:<br>• **Draft \| Standard** – is used based upon the status.<br>• **<major>** - sequential number of the major version.<br>• **<minor>** - sequential number of the minor version<br>• **<revision>** - optional sequential number of the revision. | 2 |

### 5.9.1 Major Versions

A major version of a UN/CEFACT XML Schema file constitutes significant non-backwards compatible changes. If any XML instance based on an older major

version of UN/CEFACT XML Schema attempts validation against a newer version, it may experience validation errors. A new major version will be produced when non-backward compatible changes occur. This would include the following changes:

- Removing or changing values in enumerations
- Changing of element names, type names and attribute names
- Changing the structures so as to break polymorphic processing capabilities
- Deleting or adding mandatory elements or attributes
- Changing cardinality from mandatory to optional

Major version numbers will be based on logical progressions to ensure semantic understanding of the approach and guarantee consistency in representation. Non-negative, sequentially assigned incremental integers satisfy this requirement.

| | | |
|---|---|---|
| [R 9049] | Every XML Schema file major version number MUST be a sequentially assigned incremental integer greater then zero. | 1 |

### 5.9.2 Minor Versions

The minor versioning of an XML Schema file identifies its compatibility with the preceding and subsequently minor versions within the same major version.

Within a major version of an UN/CEFACT XML Schema file there can be a series of minor, or backward compatible, changes. The minor versioning of an UN/CEFACT XML Schema file determines its compatibility with UN/CEFACT XML Schema files with preceding and subsequent minor versions within the same major version. The minor versioning scheme thus helps to identify backward and forward compatibility. Minor versions will only be increased when compatible changes occur, i.e

- Adding values to enumerations
- Optional extensions
- Add optional elements

| | | |
|---|---|---|
| [R A735] | Minor versioning MUST be limited to declaring new optional XML content, extending existing XML content, or refinements of an optional nature. | 1 |

Minor versions will be declared using the **`xsd:version`** attribute in the **`xsd:schema`** element. It is only necessary to declare the minor version in the schema version attribute since instance documents with different minor versions are compatible with the major version held in the same namespace. By using the version attribute in each document instance, the application can provide the appropriate logic switch for different compatible versions without having knowledge of the schema version which the document instance was delivered.

Minor version changes are not allowed to break compatibility with previous versions within the same major version. Compatibility includes consistency in naming of the schema constructs to include elements, attributes, and types. UN/CEFACT minor version changes will not include renaming XML Schema constructs.

975 For a particular namespace, the major version and subsequent minor versions and
976 revisions create a linear relationship.

| | | |
|---|---|---|
| [R AFA8] | Minor versions MUST NOT rename existing XML Schema defined artifacts. | 1 |
| [R BBD5] | Changes in minor versions MUST NOT break semantic compatibility with prior versions having the same major version number. | 1 |
| [R 998B] | XML Schema files for a minor version XML Schema MUST incorporate all XML Schema components from the immediately preceding version of the XML Schema file. | 1 |

977

# 6 Application of Context

The intent of the UN/CEFACT XML Schema and the NDR is to express everything that is necessary to enable integration of business information within an XML Schema conformant XML instance message. To accomplish this, the XML Schema must address all aspects of the business information to include:

- Business semantics – The meaning of business information in communication.
  - Meaning can carry between different individuals depending upon the context of the sender and receiver of the information.
  - Meaning can be the same between different individuals depending context of the sender and receiver of the information.
- Business context – The circumstances that determine the meaning of business information. The business context may change the semantic meaning for the sender and or the receiver of the information.

In CCTS, CCs are context neutral artifacts that when context is applied, result in BIE artifacts. To help standardize the process, CCTS defines different context categories that capture the different context category values. BIE artifacts and their XML Component expressions may be defined within any number of combinations of context categories and context category values. The namespace mechanism will ensure name collision of similarly named components in different contexts does not occur.

XML Schema Components representing BIE artifacts will be grouped by a single principal context category value. This principal context value will be expressed as part of the namespace to which the component is assigned. For UN/CEFACT this principal context category will be the Business Process value in which the BIE artifact is defined. Other organizations may choose to express any context category value in the namespace that fits their requirements.

How the principal context category is defined in the namespace scheme is described in section 5.6.

**Note:**

it is possible to extend the namespace described in section 5.6 for an implementation set of schemas to include a Context Identifier on the end of the namespace to express the full context of the reduced set of XML Schemas. While this Context Identifier is out side the scope of this technical specification, it is recommended that this identifier be a Univerisally Unique Identifier (UUID).

In addition to the principal context category, all other context category values for every BIE is expressed within the XML Schema definition for each XSD Schema Component as an **`xsd:appInfo`** declaration following the structure defined in section 7.5.2.

# 7 General XML Schema Definition Language Conventions

The XML Schema language has many constructs that can be used to express a model. The purpose of this section is to provide a profile of these constructs that can be used and to identify the constructs that should not be used as a result of general best practices.

This section defines rules related to general XML Schema Language Conventions:

- XML Schema Construct
- Attribute and Element Declaration
- Type Definitions
- Use of XML Schema Extension and Restriction
- Annotation

## 7.1 Overall XML Schema Structure and Rules

### 7.1.1 XML Schema Declaration

When defining an XML Schema file the first line must indicate the xml version and the encoding it uses. UN/CEFACT XML Schema will be defined UTF-8 encoding.

| | | |
|---|---|---|
| [R 8DB4] | The first line in an XML Schema file MUST contain:<br>"<?xml version="1.0" encoding="UTF-8"?>" | 1 |

Example 7-1 provides the form this information is provided.

**Example 7-1: XML Schema File Line 1 setting the XML version and encoding**

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

### 7.1.2 XML Schema File Identification and Copyright Information

After the first line there can be documentation typically in the form of **`xsd:comment`** lines. These comments are applicable to the XML Schema file.

| | | |
|---|---|---|
| [R ABD2] | Every XML Schema File MUST contain a comment that identifies its name immediately following the XML declaration. | 1 |
| [R BD41] | Every XML Schema File MUST contain a comment that identifies its owning agency, version and date immediately following the schema name comment using the format defined in Appendix B-2. | 1 |

### 7.1.3 Schema Declaration

The **`xsd:schema`** element must be declared to define an XML Schema document. The **`xsd:schema`** element includes attributes that affect how the rest of the document behaves and how XML parsers and other tools treat it. XML Schema best practice indicates:

- **elementFormDefault** be set to qualified.
- **attributeFormDefault** be set to unqualified.
- The prefix **xsd** be used to refer to the XML Schema namespace.

| | | |
|---|---|---|
| [R A0E5] | The **xsd:elementFormDefault** attribute MUST be declared and its value set to qualified. | 1 |
| [R A9C5] | The **xsd:attributeFormDefault** attribute MUST be declared and its value set to unqualified. | 1 |
| [R 9B18] | The **xsd** prefix MUST be used in all cases when referring to the namespace **http://www.w3.org/2001/XMLSchema** as follows: **xmlns:xsd=http://www.w3.org/2001/XMLSchema**. | 1 |

Example 7-2 shows a XML Schema code snippet declaring the namespace token **xsd**, setting **elementFormDefault** to qualified and setting **attributeFormDefault** to unqualified.

**Example 7-2: Element and Attribute Form Default**

```
<xsd:schema targetNamespace=" ... see namespace ...
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
```

### 7.1.4 CCTS Artifact Metadata

CCTS defines specific metadata associated with each CCTS artifact. This metadata will be expressed as a separate CCTS Metadata XML Schema File.

The CCTS XML Schema File will be named Core Components Technical Specification Schema File.

The CCTS XML Schema File will be assigned to its own namespace and use a prefix of **ccts**. The current version of this namespace is:

**urn:un:unece:uncefact:documentation:common:3:standard:CoreComponentsTechnicalSpecification**.

| | | |
|---|---|---|
| [R 90F1] | All required CCTS metadata for ABIEs, BBIEs, ASBIEs, and BDTs must be defined in an XML Schema file. | 1 |
| [R 9623] | The name of the CCTS Metadata XML Schema file will be “Core Components Technical Specification Schema File” and will be defined within the comment within the XML Schema file. | 1 |
| [R 9443] | The CCTS Metadata XML Schema File MUST reside in its own namespace which MUST be defined in accordance with rule 8E2D and assigned the prefix **ccts**. | 1 |

### 7.1.5 Constraints on Schema Construction

In addition to general XML Schema structure, best practice identifies constraints on certain XML Schema rules necessary to ensure maximum interoperability for business-to-business and application-to-application interoperability.

| | | |
|---|---|---|
| [R AD26] | **`xsd:notation`** MUST NOT be used. | 1 |
| [R ABFF] | The **`xsd:any`** element MUST NOT be used. | 4, 6 |
| [R AEBB] | The **`xsd:any`** attribute MUST NOT be used. | 4, 6 |
| [R 9859] | Mixed content MUST NOT be used. | 1 |
| [R 926D] | **`xsd:substitutionGroup`** MUST NOT be used. | 4, 6 |
| [R 8A83] | **`xsd:ID`**/**`xsd:IDREF`** MUST NOT be used. | 1 |
| [R 8E89] | **`xsd:key`**/**`xsd:keyref`** MUST be used for element referencing. | 1 |

## 7.2 Attribute and Element Declarations

### 7.2.1 Attributes

Attributes are only used in two cases:

- To convey the supplementary components of BDTs;
- To serve as identifiers and references when two elements need to be related to one another via schema identify constraints such as key-key-ref constraints.

| | | |
|---|---|---|
| [R B221] | Supplementary component information MUST be represented as Attributes. | 1 |
| [R AFEE] | User defined attributes MUST only be used for Supplementary components. | 1 |
| [R 8EE7] | Attributes MUST be used rather than elements to serve as identifiers when two elements need to be related to one another via schema identity constraints. | 1 |
| [R 9FEC] | An **`xsd:attribute`** that represents a supplementary component with variable information MUST be based on an appropriate XML Schema built-in simpleType. | 1 |
| [R B2E8] | A **`xsd:attribute`** that represents a supplementary component which uses codes MUST be based on the **`xsd:simpleType`** of | 1 |

| | | |
|---|---|---|
| | the appropriate code list. | |
| [R 84A6] | A **xsd:attribute** that represents a supplementary component which uses identifiers MUST be based on the **xsd:simpleType** of the appropriate identifier scheme. | 1 |

### 7.2.2 Elements

Elements are declared for the document level business information payload, ABIEs, BBIEs, and ASBIEs.

#### 7.2.2.1 Element Declaration

Every **ccts:BBIE** artefact is declared as an **xsd:element** of the simple or complex type that instantiates its BDT.

| | | |
|---|---|---|
| [R BCD6] | Every BBIE leaf element declaration MUST be of the BusinessDataType that represents the source basic business information entity (BBIE) data type. | 1 |

Example 7-3 shows an example declaration.

**Example 7-3: Element Declaration**

```
<xsd:complexType name="AccountType">
        <xsd:annotation>
                ...see annotation...
        </xsd:annotation>
        <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"
                        minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                ...see annotation...
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Status" type="ram:StatusType"
                        minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                ...see annotation...
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Name" type="bdt:NameType"
                        minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                ...see annotation...
                        </xsd:annotation>
                </xsd:element>
    <xsd:element name="BuyerParty" type="ram:BuyerPartyType/>
        </xsd:sequence>
</xsd:complexType>
```

#### 7.2.2.2 Empty Elements

In general, the absence of an element in an XML schema does not have any particular meaning - it may indicate that the information is unknown, or not applicable, or the element may be absent for some other reason. The XML Schema specification does provide a feature, the xsd:nillable attribute, whereby an element

may be transferred with no content, but still use its attributes and thus carry semantic meaning.

In order to respect the principles of the CCTS and to retain semantic clarity the nillability feature of XML Schema will not be used by UN/CEFACT XML Schemas.

| | | |
|---|---|---|
| [R 8337] | The **`xsd:nillable`** attribute MUST NOT be used. | 1 |

## 7.3 Type Definitions

An XML Schema Type defines simple and complex structures used to define an element.

All elements declared will have a named type that provides the definition of the structure of the XML Schema Component using it.

| | | |
|---|---|---|
| [R 8608] | Anonyms types MUST NOT be used. | 1 |

### 7.3.1 Simple Type Definitions

xsd:simpleTypes must always be used where they satisfy the user's business requirements. Where these business requirements cannot be satisfied, user defined complex type definitions will be used. Examples7-4 shows a simple type defined in the BDT XML Schema file. Example 7-5 shows a simple type defined in a Code List XML Schema file.

**Example 7-4: Simple Types in Businsess Data Type XML Schema File**

```
<xsd:simpleType name="DateTimeType">
        <xsd:annotation>
                        … see annotation …
        </xsd:annotation>
        <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
```

**Example 7-5: Simple Types in a Code Lists XML Schema File**

```
<xsd:simpleType name="CurrencyCodeContentType">
        <xsd:restriction base="xsd:token">
                <xsd:enumeration value="ADP">
                        …see enumeration of code lists …
                </xsd:enumeration>
           <xsd:annotation>
                        … see annotation …
           </xsd:annotation>
 </xsd:restriction>
</xsd:simpleType>
```

### 7.3.2 Complex Type Definitions

A complex type will be defined to express the content model of each CCTS BIE. A complex type will also be defined to express the value domain of a CCTS BDT when an XML Schema built-in data type does not convey all necessary information.

| | | |
|---|---|---|
| [R A4CE] | An **`xsd:complexType`** MUST be defined for each CCTS BIE. | 1 |

| | | |
|---|---|---|
| [R BC3C] | An **xsd:complexType** MUST be defined for each CCTS BDT that cannot be fully expressed using an **xsd:simpleType**. | 1 |

Example 7-6 shows a complex type defined for an Account ABIE.

**Example 7-6: Complex Type of Object Class "AccountType"**

```
<xsd:complexType name="AccountType">
        <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
        <xsd:sequence>
                ... see element declaration ...
        </xsd:sequence>
</xsd:complexType>
```

In order to increase consistency in use and enable accurate and complete representation of what is allowed in the design of CCTS artefacts, the **xsd:all** XML Schema Component will not be used.

| | | |
|---|---|---|
| [R A010] | The **xsd:all** element MUST NOT be used. | 1 |

## 7.4 Use of Extension and Restriction

The general philosophy is that all UN/CEFACT XML Schema Components will follow the model defined in Figure 5-2. These XML Schema Components are based on the concept that the underlying semantic structures of the CCs and BIEs are normative forms of standards that developers are not allowed to alter without coordination of appropriate TBG groups (including TBG17 - Harmonization) and ICG. As business requirements dictate, new CC artifacts and BIE artifacts will be created and represented through XML Schema Components by defining new types and elements declared as appropriate. The concept of derivation through the use of **xsd:extension** and **xsd:restriction** will only be used in limited circumstances.

It is understood that other standards organizations using this specification may have use either **xsd:extension** and/or **xsd:restriction** to define new constructs that are extended or restricted from existing constructs. While UN/CEFACT XML Schema Files will not use these other organizations may.

### 7.4.1 Extension

UN/CEFACT XML Schema Files may only use **xsd:extension** in the Business Data Type XML Schema File to declare attributes to accommodate supplementary components.

| | | |
|---|---|---|
| [R AB3F] | **xsd:extension** MUST only be used in the Business DataType XML Schema file. | 46 |
| [R 9D6E] | **xsd:extension** MUST only be used for declaring xsd:attributes to accommodate relevant supplementary components. | 46 |

### 7.4.2 Restriction

The CCTS specification employs the concept of semantic restriction in creating specific instantiations of core components. Accordingly, **xsd:restriction** will be used as appropriate to define types that are derived from the existing types. Where used, the derived types must always be renamed. Simple and complex type restrictions may be used. **xsd:restriction** can be used for facet restriction and/or attribute restriction.

| | | |
|---|---|---|
| [R 8AF7] | When **xsd:restriction** is applied to a **xsd:simpleType** or **xsd:complexType** that represents a data type the derived construct MUST use a different name. | 1 |

Example 7-7 shows a restriction of a simple type.

**Example 7-7: Restriction of Simple Type**

```
<xsd:simpleType name="TaxAmountType">
        <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
        <xsd:restriction base="bdt:AmountType">
                <xsd:totalDigits value="10"/>
                <xsd:fractionDigits value="3"/>
        </xsd:restriction>
  </xsd:simpleType>
```

## 7.5 Annotation

All UN/CEFACT XML Schema constructs will use **xsd:annotation** to provide the documentation and indicate the application of context categories specified by CCTS.

| | | |
|---|---|---|
| [R 847A] | Each defined or declared construct MUST use the **xsd:annotation** element for required CCTS documentation. | 1 |

### 7.5.1 Documentation

The annotation **xsd:documentation** will be used to convey all metadata as specified in the CCTS, i.e., to convey the semantic content carried in the XML construct. All elements specified within an **xsd:documentation** element will be expressions of ccts artifact metadata.

As identified in section 7.1.4, a CCTS Metadata XML Schema File contains definitions for all required CCTS metadata for those CCTS artifacts used in this technical specification. The CCTS Metadata XML Schema File will be imported in all Root, ABIE, Code List, and BDT schema which contain **xsd:documentation** elements.

The following annotations are required as defined in section ***Error! Reference source not found. Error! Reference source not found.*** in type definitions and element declarations (the representation of each item in XML code is shown in parenthesis):

- **UniqueID** – The unique identifier assigned to the artefact in the library. (UniqueID)
    - The UniqueID is based on EntityUniqueIdentifierType, which refers to the schema module "CCIS1 Entity Unique Identification Scheme" that provides the suggested schema pattern: "UNBE0-9{6}
- **VersionID** – The unique identifier assigned to the version of the artefact in the library.
    - The VersionID is based on VersionIdentifierType, which refers to the scheme module "CCTS4 Versioning Identification Scheme" that provides the suggested schema pattern: 0-9{1,2}\.0-9{2}
- **SequencingKeyID** – Indicates the sequence of the documentation.
- **CCTS Artefact** – The type of component. The possible values are: **RSM|BBIE|ABIE|ASBIE|BDT**
- **Name –** The name of the supplementary component or business information payload. (Name)
- **Definition** – The semantic meaning of the artefact. (Definition)
    - The Definition is based on BDT "TextType". The language representation should follow the same approach as described for name.
- **Cardinality** – Indicates the cardinality of the documentation.
- **Object Class Qualifier Name** – A name that qualifies the Object Class.
- **Object Class Name** – The Object Class represented by the artefact.
- **PropertyQualifier Name** – The name of the property qualifier.
- **PropertyTermName** – The name of the property.
- **RepresentationTermName** – The name of the representation term.
- **UsageRule** – Indicates the Usage Rule of the Object.
- **BusinessTermName –** A synonym term under which the artefact is commonly known and used in business. (BusinessTerm)

Appendix F specifies normative information on the specific annotation required for each of the artifacts.

Example 7-8 provides an example of annotation documentation for a BBIE that conforms to the ccts structure.

**Example 7-8: Example of Annotation Documentation**

```
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID>UNBE000000</ccts:UniqueID>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:SequencingKeyID>1</ccts:SequencingKeyID>
    <ccts:CCTSArtifact>BBIE</ccts:CCTSArtifact>
    <ccts:DictionaryEntryName>Account. Identifier</ccts:DictionaryEntryName>
    <ccts:Definition> The identification of a specific account.</ccts:Definition>
    <ccts:Cardinality>String</ccts:Cardinality>
```

```
    <ccts:ObjectClassName>Account<ccts:ObjectClassName>
    <ccts:PropertyTermName>Identifier<ccts:PropertyTermName>
    <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
  </xsd:documentation>
</xsd:annotation>
```

Each UN/CEFACT construct containing a code must include documentation that will identify the code list(s) that must be supported when the construct is used.

Example 7-9 shows the XML Schema definition of annotation documentation for each of the type of component.

**Example 7-9: XML Schema definition of annotation documentation**

```
<xsd:schema

xmlns:ccts="urn:un:unece:uncefact:documentation:common:3:standard:XMLNDRDocumentati
on"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:bdt="urn:un:unece:uncefact:data:common:3:standard:BusinessDataType"
   targetNamespace="
urn:un:unece:uncefact:documentation:common:3:standard:XMLNDRDocumentation"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:import namespace="
urn:un:unece:uncefact:data:common:3:standard:BusinessDataType"

schemaLocation="http://www.unece.org/uncefact/data/common/3/standard/BusinessDataTy
pe_3p0.xsd"/>
   <xsd:group name="RootSchema_Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="ABIE_Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="BBIE Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="ASBIE_Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="BDT_Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="BDT_SC_Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="CodeList_Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:group name="CodeValue Documentation">
      <xsd:sequence>
         ....
      </xsd:sequence>
   </xsd:group>
   <xsd:complexType name="UsageRuleType">
      <xsd:sequence>
                  <xsd:element name="UniqueID" type="bdt:IDType" minOccurs="0"/>
                  <xsd:element name="Name" type="bdt:NameType" minOccurs="0"/>
```

```
                        <xsd:element name="Description" type="bdt:TextType"/>
                        <xsd:element name="Constraint" type="bdt:TextType"/>
                        <xsd:element name="ConstraintTypeCode" type="bdt:ConstraintTypeCodeType"
minOccurs="0"/>
                        <xsd:element name="ConditionTypeCode" type="bdt:ConditionTypeCodeType"
minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    ....
</xsd:schema>
```

Table 7-1 provides a summary view of the annotation data as defined in this section.

| | rsm:RootSchema | ABIE xsd:complexType | BBIE xsd:element | ASBIE: xsd:element | cct:CoreComponentTyp e | supplementary component | cdt:CoreDataType | bdt:BusinessDataType |
|---|---|---|---|---|---|---|---|---|
| Unique Identifier | M | M | M | M | M | | M | M |
| CCTS Artifact | M | M | M | M | M | M | M | M |
| Dictionary Entry Name | | M | M | M | M | | M | M |
| Name | M | | | | | M | | |
| Version Identifier | M | M | M | M | M | | M | M |
| Definition | M | M | M | M | M | M | M | M |
| Cardinality | | | M | M | | M | | |
| Object Class Term Name | | M | M | M | | M | | |
| Object Class Qualifier Term Name | | O | O | O | | | | |
| Property Term Name | | | M | M | | M | | |
| Property Qualifier Term Name | | | O | O | | | | |
| Associated Object Class Term Name | | | | M | | | | |
| Associated Object Class Qualifier Term Name | | | | O | | | | |
| Association Type | | | | M | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Primary Representation Term Name | | | M | | M | | | M |
| Data Type Qualifier Term Name | | | | | | | | M |
| Primitive Type Name | | | | | M | M | M | M |
| Usage Rule | | O, R | O, R | O, R | O, R | | O, R | O, R |
| Business Term Name | | O, R | O, R | O, R | O, R | | | |
| Example | | O, R | O, R | O, R | O, R | O, R | O, R | O, R |

Key: M – Mandatory O – Optional R – Repeating C – Conditional

**Table 7-1 Annotation Data Summary**

#### 7.5.1.1 Usage Rules

CCTS defines the concept of usage rules to convey instructions on how to use a CCTS artifact in a given context. These usage rules have a **`ccts:ConstraintType`** which classifies the rules as being structured (expressed in a formal language such as the Object Management Group's Object Constraint Language (OCL)) or unstructured (free form text).

**Note:**

The UN/CEFACT TMG UCM project is defining the context mechanism that will support refining usage rules in a given business circumstance. Once that specification is finalized, this section may change.

##### 7.5.1.1.1 Structured Usage Rules

Structured usage rules are suitable for direct application processing and will communicated as part of an XML Schema Component through the **`xsd:documentation`** element.

| | | |
|---|---|---|
| [R 88DE] | Usage rules whose **`ccts:ConstraintType`** is something other than "**`unstructured`**" MUST be expressed within a **`ccts:UsageRule`** element within an **`xsd:documentation`** element. | 1 |
| [R B851] | The structure of the **`ccts:ConstraintType`** element MUST be:<br>• ccts:UniqueID [1..1]<br>• ccts:Constraint [1..1]<br>• ccts:ConstraintType [1..1]<br>• ccts:ConditionType [1..1 | 1 |

| | | |
|---|---|---|
| | • ccts:Name [0..1]<br>• ccts:BusinessTerm [0..*] | |

1353 #### 7.5.1.1.2 Unstructured Usage Rules

1354 Unstructured usage rules are not suitable for direct application processing and will
1355 communicated as part of an XML Schema Component through the
1356 **`xsd:documentation`** element.

| | | |
|---|---|---|
| [R A1CF] | Usage rules whose **`ccts:ConstraintType`** is unstructured MUST be expressed within a **`ccts:UsageRule`** element within an **`xsd:documentation`** element. | 1 |

1357 ## 7.5.2 Application Information (AppInfo)

1358 The annotation **`xsd:appInfo`** will be used to convey the context specified that is
1359 applicable for all BIE artifacts and the resulting XML Schema Components used to
1360 express them. All context categories will be expressed using the CCTS context
1361 category structures that are defined as shown in Example 7-10.

1362 All elements specified within an **`xsd:appInfo`** element will be expressions of CCTS
1363 context categories.

1364 As identified in section 7.1.4, a CCTS Metadata XML Schema File contains
1365 definitions for all required CCTS metadata and contexts for CCTS artifacts used in
1366 this technical specification. The CCTS Metadata XML Schema File is imported in all
1367 Root, ABIE, Code List, and BDT schema which contain **`xsd:appInfo`** elements.

1368 The following **`xsd:appInfo`** structures are defined and used as described in section
1369 ***Error! Reference source not found. Error! Reference source not found.*** in type
1370 definitions and element declarations. The BusinessContext defined within each
1371 xsd:appInfo contains one or more ccts:ContextUnit which contains each of the
1372 identified context categories recognized by CCTS.

1373 • Business Process Context Category
1374 • Business Process Role Context Category
1375 • Supporting Role Context Category
1376 • Industry Classification Context Category
1377 • Product Classification Context Category
1378 • Geopolitical Context Category
1379 • Official Constraints Context Category
1380 • System Capabilities Context Category

1381 Example 7-10 shows the XML Schema definition of annotation appInfo structures
1382 which start with BusinessContext that is to be applied for each of the XML Schema
1383 Components element, complexType and simpleType.

1384 **Example 7-10: XML Schema definition of annotation appInfo**

```
<xsd:schema

xmlns:ccts="urn:un:unece:uncefact:documentation:common:3:standard:XMLNDRDocumentati
on"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:bdt="urn:un:unece:uncefact:data:common:3:standard:BusinessDataType"
   targetNamespace="
urn:un:unece:uncefact:documentation:common:3:standard:XMLNDRDocumentation"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:import namespace="
urn:un:unece:uncefact:data:common:3:standard:BusinessDataType"

schemaLocation="http://www.unece.org/uncefact/data/common/3/standard/BusinessDataTy
pe_3p0.xsd"/>
  ....
   <xsd:element name="BusinessContext">
            <xsd:complexType>
                     <xsd:sequence>
                              <xsd:element name="ContextUnit" maxOccurs="unbounded">
                                       <xsd:complexType>
                                                <xsd:sequence>
                                                         <xsd:element
name="BusinessProcessContextCategory" type="ccts:BusinessProcessContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                         <xsd:element
name="BusinessProcessRoleContextCategory" type="ccts:BusinessProcessRoleContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                         <xsd:element
name="SupportingRoleContextCategory" type="ccts:SupportingRoleContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                         <xsd:element
name="IndustryClassificationContextCategory" type="ccts:IndustryClassificationContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                         <xsd:element
name="ProductClassificationContextCategory" type="ccts:ProductClassificationContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                         <xsd:element name="GeopoliticalContextCategory"
type="ccts:GeopoliticalContextCategoryType" minOccurs="0" maxOccurs="unbounded"/>
                                                         <xsd:element
name="OfficialConstraintsContextCategory" type="ccts:OfficialConstraintsContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                         <xsd:element
name="SystemCapabilitiesContextCategory" type="ccts:SystemCapabilitiesContextCategoryType" minOccurs="0"
maxOccurs="unbounded"/>
                                                </xsd:sequence>
                                       </xsd:complexType>
                              </xsd:element>
                     </xsd:sequence>
                     <xsd:attribute name="id" type="bdt:EntityUniqueIdentifierType"/>
                     <xsd:attribute name="versionID" type="bdt:VersionIdentifierType"/>
            </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="BusinessInformationContextCategoryType">
            <xsd:sequence>
                     <xsd:element name="BusinessInformationEntityID" type="bdt:IDType"
maxOccurs="unbounded"/>
                     <xsd:element name="ContextExclusion" minOccurs="0">
                              <xsd:complexType>
                                       <xsd:sequence>
                                                <xsd:element name="BusinessInformationEntityID"
type="bdt:IDType" maxOccurs="unbounded"/>
                                       </xsd:sequence>
                              </xsd:complexType>
                     </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
   </xsd:complexType>
   <xsd:complexType name="BusinessProcessContextCategoryType">
            <xsd:sequence>
                     <xsd:element name="BusinessProcessCode" minOccurs="0" maxOccurs="unbounded">
                              <xsd:complexType>
                                       <xsd:complexContent>
                                                <xsd:extension base="bdt:CodeType"/>
                                       </xsd:complexContent>
                              </xsd:complexType>
```

```
                        </xsd:element>
                        <xsd:element name="ContextExclusion" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="BusinessProcessTypeCode"
type="bdt:CodeType" maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
        </xsd:complexType>
        <xsd:complexType name="BusinessProcessRoleContextCategoryType">
                <xsd:sequence>
                        <xsd:element name="BusinessProcessRoleCode" type="bdt:CodeType" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element name="ContextExclusion" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="PartyFunctionCode" type="bdt:CodeType"
maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
        </xsd:complexType>
        <xsd:complexType name="SupportingRoleContextCategoryType">
                <xsd:sequence>
                        <xsd:element name="SupporterFunctionCode" minOccurs="0" maxOccurs="unbounded">
                                <xsd:complexType>
                                        <xsd:complexContent>
                                                <xsd:extension base="bdt:CodeType"/>
                                        </xsd:complexContent>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="ContextExclusion" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="SupporterFunctionCode"
type="bdt:CodeType" maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
        </xsd:complexType>
        <xsd:complexType name="IndustryClassificationContextCategoryType">
                <xsd:sequence>
                        <xsd:element name="IndustryClassificationCode" type="bdt:CodeType" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element name="ContextExclusion" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="IndustryTypeCode" type="bdt:CodeType"
maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
                <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
        </xsd:complexType>
        <xsd:complexType name="ProductClassificationContextCategoryType">
                <xsd:sequence>
                        <xsd:element name="ProductClassificationCode" type="bdt:CodeType" minOccurs="0"
maxOccurs="unbounded"/>
                        <xsd:element name="ContextExclusion" minOccurs="0">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="ProductTypeCode" type="bdt:CodeType"
maxOccurs="unbounded"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:sequence>
```

```
            <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
   </xsd:complexType>
   <xsd:complexType name="GeopoliticalContextCategoryType">
            <xsd:sequence>
                  <xsd:element name="GeopoliticalCode" minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="ContextExclusion" minOccurs="0">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element ref="clm54217:CurrencyCode"
maxOccurs="unbounded"/>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
   </xsd:complexType>
   <xsd:complexType name="OfficialConstraintsContextCategoryType">
            <xsd:sequence>
                  <xsd:element name="OfficialConstraintsCode" minOccurs="0" maxOccurs="unbounded">
                        <xsd:complexType>
                              <xsd:complexContent>
                                    <xsd:extension base="bdt:CodeType"/>
                              </xsd:complexContent>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="ContextExclusion" minOccurs="0">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="LawTypeCode" type="bdt:CodeType"
maxOccurs="unbounded"/>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="inAllContextsListIndicator" type="xsd:boolean"/>
   </xsd:complexType>
   <xsd:complexType name="SystemCapabilitiesContextCategoryType">
            <xsd:sequence>
                  <xsd:element name="SystemCapabilitiesID" type="bdt:IDType" minOccurs="0"
maxOccurs="unbounded"/>
                  <xsd:element name="ContextExclusion" minOccurs="0">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SoftwareSolutionID" type="bdt:IDType"
maxOccurs="unbounded"/>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="inAllContextsIndicator" type="xsd:boolean"/>
   </xsd:complexType>
</xsd:schema>
```

Using this structure it is possible to indicate all of the context categories in which a BIE is applicable.

Example 7-11 shows a generic example of using the structures for appInfo to communicate the context categories in which a given element is applicable.

**Example 7-11 Use of the `xsd:appInfo` Business Context**

```
<xsd:element name="<name>" type="<type>">
   <xsd:annotation>
   … (documentation) …
   <xsd:appinfo source="urn:un:unece:uncefact:businesscontext….">
            <ccts:BusinessContext>
                  <ccts:ContextUnit>
                        <ccts:BusinessProcessContextCategory>
                              <ccts:BusinessTransactionDocumentCode>0062
                              </ccts:BusinessTransactionDocumentCode>
                              <!-- PurchasingContractUseRequest -->
                              <ccts:BusinessTransactionDocumentCode>0081
```

```
                                    </ccts:BusinessTransactionDocumentCode>
                                    <!-- CataloguePublicationRequest -->
                                    … (further business transaction document codes) ….
                             </ccts:BusinessProcessContextCategory>
                             <ccts:IndustryClassificationContextCategory>
                                    <ccts:IndustryClassificationCode>0001
                                    </ccts:IndustryClassificationCode>
                                    <!-- Aerospace -->
                                    <ccts:IndustryClassificationCode>0002
                                    </ccts:IndustryClassificationCode>
                                    <!-- Defence -->
                                    <ccts:IndustryClassificationCode>0006
                                    </ccts:IndustryClassificationCode><!– CP  -->
                                    … (further business transaction document codes) ….
                             </ccts:IndustryClassificationContextCategory>
                             <ccts:GeopoliticalContextCategory>
                                    <ccts:CountryCode>DE</ccts:CountryCode>
                                    <!-- Germany -->
                                    <ccts:CountryCode>FR</ccts:CountryCode>
                                    <!-- France -->
                                    <ccts:CountryCode>US</ccts:CountryCode>
                                    <!-- USA -->
                                    <ccts:CountryCode>AT</ccts:CountryCode>
                                    <!-- Austria -->
                                    … (further business transaction document codes) ….
                             </ccts:GeopoliticalContextCategory>
                             … (further business context categories) ….
                      <ccts:ContextUnit>
               </ccts:BusinessContext>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:element>
```

**Note:**

The UN/CEFACT TMG UCM project is defining the context mechanism that will support refining context categories in a given business circumstance. Once that specification is finalized, this section may change.

# 8 Application of Context in Namespace

As indicated in *5.7 XML Schema Files* the XML Schema files have dependencies upon one another. Figure 8-1 further shows these dependencies and shows how these dependencies are realized using **`xsd:include`** and **`xsd:import`**. Furthermore one of the context categories are implemented within the namespace scheme all of the XML Schema files for the given value of that context category are defined within the corresponding namespace. The XML Schema files for other values of the context category are defined in namespaces corresponding to those values.

[Note]

It is important to note here again that UN/CEFACT has chosen to implement the Business Process context category in the namespace.

Figure 8-1 shows two context category values “A” and “B.” These namespace are independently declared and may not have any shared dependencies other than to common Code Lists that are themselves independent of context.

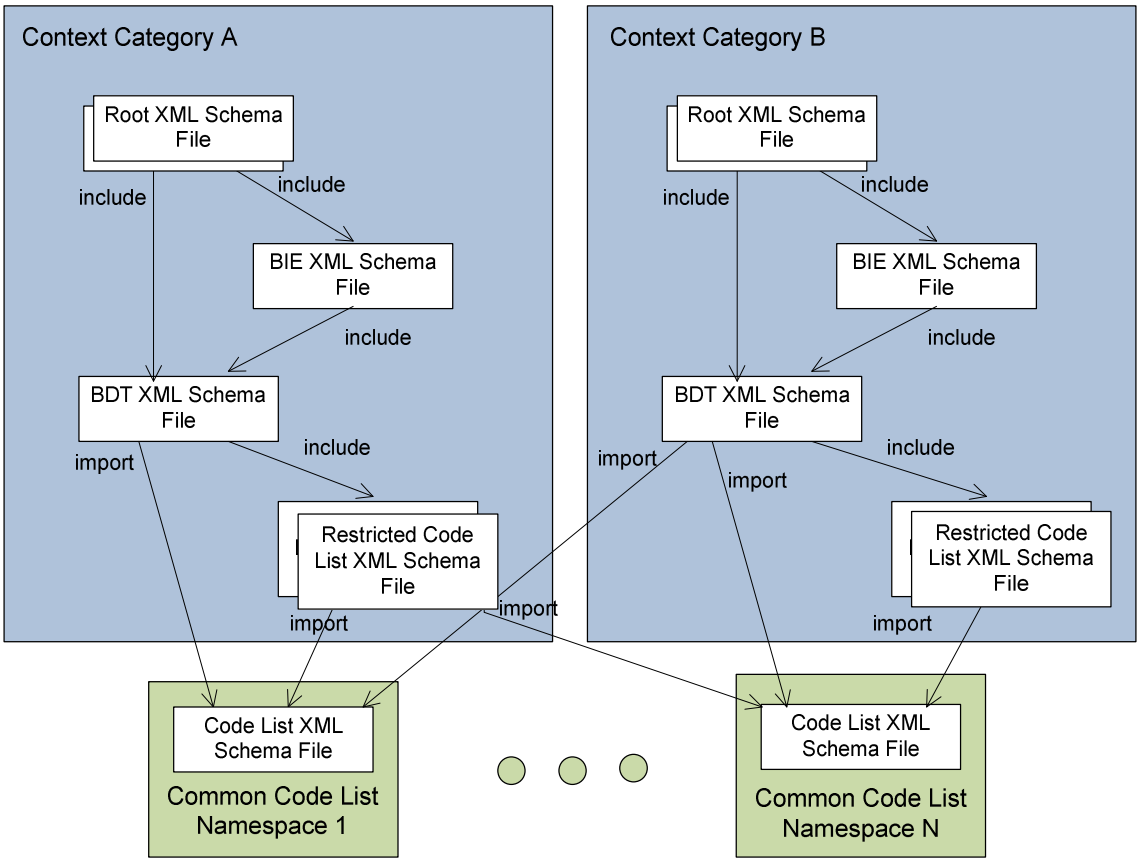



**Figure 8-1: Imports and Includes of XML Schema Files for UN/CEFACT Modularity Model**

All XML Schemas published by UN/CEFACT will be assigned to a unique namespace and token by ATG that represents business process context category in which it is designed.

| | | |
|---|---|---|
| [R B96F] | The Root XML Schema file MUST be assigned to a unique namespace token that represents the context category value it is intended. | 1 |

Example 8-1 is an example of a namespace for the context category business process value Order Management.

**Example 8-1: Namespace for Context Category Business Process – Order Management**

```
"xmlns:ordman=”urn:un:unece:uncefact:oredermanagement:data:draft:1"
```

Example 8-2 shows how a given XML Schema file is declared to be within the context category business process value Order Management.

**Example 8-2: Schema-Element of an XML Schema File within the Context Category Business Process Value – Order Management**

```
<xsd:schema
  targetNamespace=
  "urn:un:unece:uncefact:ordermangement:data:1:draft"
  xmlns:ordman=
  "urn:un:unece:uncefact:ordermanagement:data:1:draft"
```

[Note]

Implementations of this specification require the implementation to use a namespace prefix like **`ordman`** for the Business Process – Order Management

This section further describes the requirements of the application of context in the namespace of the various XML Schema files that are incorporated within the UN/CEFACT library.

- Root XML Schema Files
- Business Information Entities XML Schema File
- Business Data Type XML Schema File
- Code List XML Schema Files
  - General Code List Contructs
  - Restricted Code List XML Schema Files
  - Common Code List XML Schema Files

## 8.1 Root XML Schema Files

The Root XML Schema file serves as the container for all schema defined content that is required to fulfill a business information exchange for the given business information payload for the context category expressed in the namespace. All of the Root XML Schema files that are necessary to fulfill the context category are defined within the namespace that is defined by that context category value.

Figure 8-1 shows multiple Root XML Schema files defined in the two context category based namespaces. The number of Root XML Schema files for a given context category may be 1 or more.

### 8.1.1 XML Schema Structure

Each Root XML Schema file will be structured in a standardized format in order to ensure consistency and ease of use. The specific format is shown in Example 8-3. The Root XML Schema file must adhere to the format of the relevant sections as detailed in Appendix B.

**Example 8-3: Structure of Root XML Schema File**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ========================================================================= -->
<!-- =====  [MODULENAME] XML Schema File                                  ===== -->
<!-- ========================================================================= -->
<!--
   Schema agency:        UN/CEFACT
   Schema version:       3.0
   Schema date:          15 July 2008

  Copyright (C) UN/CEFACT (2008). All Rights Reserved.

... see copyright information ...
-->
<xsd:schema
  targetNamespace="urn:un:unece:uncefact:data:ordermanagement:3:draft"
  ... see namespaces ...
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified" version="3.0">
  <!-- ========================================================================= -->
  <!-- =====  Imports                                                    ===== -->
  <!-- ========================================================================= -->
  <!-- =====  Import of [MODULENAME]                                     ===== -->
  <!-- ========================================================================= -->
  <   see imports
  <!-- ========================================================================= -->
  <!-- =====  Include                                                    ===== -->
  <!-- ========================================================================= -->
  <!-- ===== Include of [MODULENAME]                                   ===== -->
  <!-- ========================================================================= -->
  ... see includes ...
  <!-- ========================================================================= -->
  <!-- =====  Element Declarations                                       ===== -->
  <!-- ========================================================================= -->
  <!-- =====  Root Element Declarations                                  ===== -->
  <!-- ========================================================================= -->
     See element declarations…
  <!-- ========================================================================= -->
  <!-- =====  Type Definitions                                           ===== -->
  <!-- ========================================================================= -->
  <!-- =====  Type Definitions: [TYPE]                                   ===== -->
  <!-- ========================================================================= -->
  <xsd:complexType name="[TYPENAME]">
         <xsd:restriction base="xsd:token">
                ... see type definition ....
         </xsd:restriction>
  </xsd:complexType>
</xsd:schema>
```

### 8.1.2 Includes

As shown in Figure 8-1 within the namespace for context category one or more Root XML Schema files will include the BIE XML Schema file and the BDT XML Schema file that reside in the same namespace.

XML Schema files in one context category specific namespace must not import XML Schema file in another context specific namespace. Since the contexts of these namespaces are not dependent upon neither should the XML Schema files be dependent upon one another. If however the a valid context can be defined such that the context applies to all of the Root Schemas these XML Schema file must share BIE and BDT XML Schema files.

| | | |
|---|---|---|
| [R B698] | The Root XML Schema file MUST include the XML Schema files that are in the same namespace as the Root XML Schema file:<br>• BIE XML Schema file<br>• BDT XML Schema file | 1 |
| [R ACBD] | A Root Schema in one namespace that is dependent upon type definitions or element declarations defined in another namespace MUST NOT import XML Schema Files from that namespace. | 1 |

### 8.1.3 Root Element Declaration

Each UN/CEFACT business information payload message has a single root element that is globally declared in the Root XML Schema File. The global element is named according to the business information payload that it represents and references the target information payload that contains the actual business information.[4]

| | | |
|---|---|---|
| [R BD9F] | A global element known as the root element, representing the business information payload, MUST be declared in the Root XML Schema File. | 1 |
| [R A466] | The name of the root element MUST be the name of the business information payload with separators and spaces removed. | 1 |
| [R 8062] | The root element declaration MUST be defined using **`xsd:complexType`** that represents the definition of the business information payload. | 1 |

Example 8-4 shows an example declaration of a Root Element.

**Example 8-4: Declaration of Root Element**

```
<!-- ================================================================== -->
<!-- ===== Root Element                                          ===== -->
<!-- ================================================================== -->
       <xsd:element name="Invoice" type="rsm:InvoiceType">
       <xsd:annotation>
              ... see annotation ...
       </xsd:annotation>
</xsd:element>
```

[4] All references to root element represent the globally declared element in a UN/CEFACT schema module that is designated as the root element for instances that use that schema.

### 8.1.4 Type Definitions

Root schemas are limited to defining a single **xsd:complexType** and a declaring a single global element that fully describe the business information payload.

| | | |
|---|---|---|
| [R 8837] | Each Root XML Schema File MUST define a **xsd:complexType** that fully describes the business information payload. | 1 |
| [R 9119] | The name of the root schema **xsd:complexType** MUST be the name of the root element with the word '**Type**' appended. | 1 |

Example 8-5 shows the definition of a complex type.

**Example 8-5: Name of Complex Type Definition**

```
<!-- ============================================================== -->
<!-- ===== Root Element                                       ===== -->
<!-- ============================================================== -->
<xsd:element name="Invoice" type="rsm:InvoiceType">
  <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
  </xsd:element>
  <xsd:complexType name="InvoiceType">
        <xsd:sequence>
                  ...
                </xsd:sequence>
  </xsd:complexType>
```

### 8.1.5 Declaration of the Referencing Constraints

Referencing between ABIEs occur in the boundaries of a particular 'scoping' element in the XML document tree (*scoping element* means an element in the hierarchy of the XML document under which a closed set of references can be defined). Most often the scoping element will be the message root element but it can also be another element lower in the hierarchy. The schema language requires that the identity constraints be defined within that scoping element.

The following principles are taken into account for the implementation of key-keyref constraints:

- For maximum element and type reuse and to stay away from forward compatibility problems, attributes used as identifiers or references are optional. This means that no **xsd:key** constraints should be defined on identifiers, which would make the identifiers mandatory in the context of a message; only **xsd:unique** constraints should be used.
- Only the ABIEs that are part of a logical aggregation implemented by XML referencing will be subject to explicit schema identity constraints. For all other ABIEs - which may only be involved in dynamic references - uniqueness of identifiers should be granted by use of adequate algorithms for the generation of the identifiers.

The identifier attribute of each ABIE that is part of a logical aggregation implemented by XML referencing will be subject to a **xsd:unique** constraint defined in the

constraint scoping element. The name of the **xsd:unique** constraint must be unique in the schema.

| | | |
|---|---|---|
| [R BA43] | For each referenced ABIE element one **xsd:unique** constraint involving the identifier attribute of the referenced element MUST be declared in the schema, under the scoping element. | 1 |
| [R B40C] | The name of the **xsd:unique** constraint MUST be composed as follows:<br>"**<Scoping Element Name Text><Referenced Element Name Text>Key**"<br>So that the name is unique in the schema. This declaration will guarantee uniqueness of the identifier attribute values across all referenced elements of the same name, in the given scope.<br>Where:<br>• Scoping Element Name Text – is the element name within XML document hierarchy which a closed set of reference is defined.<br>• Referenced Element Name Text – is the element name within the scoping element being referenced. | 1 |

In Example 8-6 the declaration under the message root element will guarantee uniqueness of the **@key** attribute values across all **ram:Party** elements, in the scope of the **rsm:ClaimNotify** message.

**Example 8-6: Unique Declaration**

```
<xsd:unique name="ClaimNotifyPartyKey">
  <xsd:selector xpath="ram:Party"/>
  <xsd:field xpath="@key"/>
</xsd:unique>
```

**Note:** The value of **xsd:selector/@xpath** identifies instances of one element in one namespace (by default the root namespace). Referenced elements defined in the data namespace need to wear the proper namespace prefix.

For each referenced ABIE used in a given scope within the message, a **xsd:keyRef** declaration must be made. Since the schema will specify which parent element can contain the reference attribute, there MUST be only one **xsd:keyRef** declaration for all the instances where the reference attribute appears.

| | | |
|---|---|---|
| [R AC2D] | For each referenced element in a given scope one **xsd:keyref** constraint involving the reference attribute that point to the referenced element MUST be declared in the XML Schema, under the scoping element. | 1 |
| [R 9BE8] | Since the XML Schema will specify which parent element can contain the reference attribute, there MUST only be one **xsd:keyref** constraint declared for all the elements where the | 1 |

| | | |
|---|---|---|
| | reference attribute may occur. | |
| [R 858D] | The name of the **xsd:keyref** constraint MUST conventionally be composed as follows:<br>"**<Scoping Element Name Text><Reference Attribute Name Text>**"<br>So that the name is unique in the schema where:<br>• Scoping Element Name Text – is the element name within XML document hierarchy which a closed set of reference is defined.<br>• Reference Attribute Name Text – is the element name within the scoping element being referenced. | 1 |

In Example 8-7 the declaration under the message root element will enforce referencing between all the elements that have the **@PartyReference** attribute and instances of **ram:Party**, in the scope of the **rsm:ClaimNotify** message.

**Example 8-7: Key Reference Declaration**

```
<xsd:keyref name="ClaimNotifyPartyReference" refer="ClaimNotifyPartyKey">
  <xsd:selector xpath=".//*"/>
  <xsd:field xpath="@partyReference"/>
</xsd:keyref>
```

**Note:**

The value of **xsd:selector/@xpath** allows for any element in any namespace to be the parent element of the reference attribute in the **xsd:keyref** constraint.

Dynamic referencing does not require the schema to enforce uniqueness of **@key** attributes when they are not involved in structural referencing. This will avoid unnecessary complexity of the identity constraints.

| | | |
|---|---|---|
| [R 886A] | Uniqueness of **@key** attributes that are not involved in structural referencing MUST NOT be enforced by the schema via identity constraints. Uniqueness of **@key** attributes should be assured by use of adequate algorithms for the generation of the identifiers (e.g. UUIDs). | 1 |

## 8.1.6 Annotations

### 8.1.6.1 Annotation Documentation

In the Root XML Schema File the root element declaration must have a structured set of annotation documentation.

| | | |
|---|---|---|
| [R 8010] | The Root XML Schema File root element declaration MUST have a structured set of annotations documentation present in that includes:<br>• UniqueID (mandatory): The identifier that references the | 1 |

| | | |
|---|---|---|
| | business information payload instance in a unique and unambiguous way.<br>• VersionID (mandatory): The identifier that reference the version of the business information payload instance.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be RSM.<br>• Name (mandatory): The name of the business information payload.<br>• Definition (mandatory): A brief description of the business information payload.<br>• BusinessTermName (mandatory): The business term name that the payload object is known by. | |

Example 8-8 shows the definition of the annotation documentation.

**Example 8-8: The annotation documentation definition for the root element documentation.**

```
<xsd:group name="RootSchemaDocumentation">
         <xsd:sequence>
                  <xsd:element name="UniqueID"
type="bdt:EntityUniqueIdentifierType"/>
                  <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                  <xsd:element name="CCTSArtifact"
type="bdt:DocumentationCCTSArtifactType" fixed="RootSchema"/>
                  <xsd:element name="Name" type="bdt:NameType"/>
                  <xsd:element name="Definition" type="bdt:TextType"/>
                  <xsd:element name="BusinessTermName" type="bdt:NameType"
minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
  </xsd:group>
```

#### 8.1.6.2 Annotation Application Information (AppInfo)

The annotation **`xsd:appInfo`** on the Root Element is used to convey the context that is applicable for the Root Element. The structure of the context is provided in section 7.5.2, Application Information (AppInfo). All contexts in which the Root Element is applicable is expressed here.

## 8.2 Business Information Entities XML Schema Files

A UN/CEFACT BIE XML Schema file is a XML Schema definition that contains all of the reusable ABIEs for the context category that is reflected in the namespace. This XML Schema file will be used (included into) in all of the UN/CEFACT Root XML Schema Files for the context category in which it is defined.

### 8.2.1 Schema Structure

Each BIE XML Schema file will be structured in a standardized format in order to ensure consistency and ease of use. The specific format is shown in Example 8-9 below and must adhere to the format of the relevant sections as detailed in Appendix B.

**Example 8-9: Structure of BIE XML Schema Files**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ========================================================================= -->
<!-- =====  ABIEs XML Schema File                              ===== -->
<!-- ========================================================================= -->
<!--
  Schema agency:         UN/CEFACT
  Schema version:        3.0
  Schema date:           15 July 2008

  Copyright (C) UN/CEFACT (2008). All Rights Reserved.

         ... see copyright information ...
-->
<xsd:schema
  targetNamespace=
  ... see namespace declaration ...
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <!-- ========================================================================= -->
  <!-- ===== Imports                                                    ===== -->
  <!-- ========================================================================= -->
  ... see imports ...
  <!-- ========================================================================= -->
  <!-- ===== Type Definitions                                           ===== -->
  <!-- ========================================================================= -->
  ... see type defintions ...
</xsd:schema>
```

## 8.2.2 Includes

The BIE XML Schema file will include the corresponding BDT XML Schema file that resides in the same namespace.

| | | |
|---|---|---|
| [R 8FE2] | The Business Information Entity XML Schema file MUST include the Business Data Type XML Schema File that resides in the same namespace. | 1 |

Example 8-10 shows the syntax for including the BDT XML Schema file.

**Example 8-10: Import of required modules**

```
  <!-- ========================================================================= -->
  <!-- ===== Includes                                                   ===== -->
  <!-- ========================================================================= -->
  <!-- ===== Include of Business Data Type XML Schema File              ===== -->
  <!-- ========================================================================= -->
  <xsd:include
  schemaLocation="http://www.unece.org/uncefact/data/ordermanagement/3/draft/Busine
ssDataType_1p0.xsd"/>
xsd"/>
```

## 8.2.3 Type Definitions

For every complex type definition based on an ABIE object class, its XSD content model will be defined such that it reflects each property of the object class as an element declaration, with its cardinality and sequencing within the XML Schema content model determined by the details of the source business information entity (ABIE).

| | | |
|---|---|---|
| [R AF95] | For every object class (ABIE) identified in the corresponding syntax-neutral model, a named **`xsd:complexType`** MUST be defined. | 1 |
| [R 9D83] | The name of the ABIE **`xsd:complexType`** MUST be the **`ccts:DictionaryEntryName`**: with the spaces and separators removed, approved abbreviations and acronyms applied and with the '**`Details`**' suffix replaced with '**`Type`**'. | 1 |
| [R 9C70] | Every aggregate business information entity (ABIE) **`xsd:complexType`** definition content model MUST use zero or more **`xsd:sequence`** and/or zero or more **`xsd:choice`** elements to reflect each property (BBIE or ASBIE) of its class. | 1 |
| [R 81F0] | Repeating series of only **`xsd:sequence`** MUST NOT occur. | 1 |
| [R 8FA2] | Repeating series of only **`xsd:choice`** MUST NOT occur. | 1 |
| [R 90F9] | The order and cardinality of the elements within an ABIE **`xsd:complexType`** MUST be according to the structure of the ABIE as defined in the model. | 1 |

No complex type may contain a sequence followed by another sequence or a choice followed by another choice, as show in Example 8-11 and Example 8-12. However, it is permissible to alternate sequence and choice as in Example 8-13.

**Example 8-11: Sequence within an object class**

```
<xsd:complexType name="AccountType" >
        <xsd:annotation>
                ...see annotation...
        </xsd:annotation>
        <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"
                        minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                ...see annotation...
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Status" type="ram:StatusType"
                        minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                ...see annotation...
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Name" type="bdt:NameType"
                        minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                ...see annotation...
                        </xsd:annotation>
                </xsd:element>
                ...
        </xsd:sequence>
</xsd:complexType>
```

**Example 8-12: Choice**

```
<xsd:complexType name="LocationType">
```

```
        <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
        <xsd:choice>
                <xsd:element name="GeoCoordinate" type="ram:GeoCoordinateType"
                        minOccurs="0">
                        <xsd:annotation>
                                ... see annotation ...
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Address" type="ram:AddressType"
                        minOccurs="0">
                        <xsd:annotation>
                                ... see annotation ...
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Location" type="ram:LocationType"
                        minOccurs="0">
                        <xsd:annotation>
                                ... see annotation ...
                        </xsd:annotation>
                </xsd:element>
        </xsd:choice>
  </xsd:complexType>
```

**Example 8-13: Sequence + Choice within Object Class "PeriodType"**

```
  <xsd:complexType name="PeriodType">
        ...
        <xsd:sequence>
                <xsd:element name="DurationDateTime"
                        type="qdt:DurationDateTimeType" minOccurs="0"
                        maxOccurs="unbounded">
                        ...
                </xsd:element>
                ...
                <xsd:choice>
                        <xsd:sequence>
                                <xsd:element name="StartTime" type="bdt:TimeType"
                                        minOccurs="0">
                                        ...
                                </xsd:element>
                                <xsd:element name="EndTime" type="bdt:TimeType"
                                        minOccurs="0">
                                        ...
                                </xsd:element>
                        </xsd:sequence>
                        <xsd:sequence>
                                <xsd:element name="StartDate" type="bdt:DateType"
                                        minOccurs="0">
                                        ...
                                </xsd:element>
                                <xsd:element name="EndDate" type="bdt:DateType"
                                        minOccurs="0">
                                        ...
                                </xsd:element>
                        </xsd:sequence>
                        <xsd:sequence>
                                <xsd:element name="StartDateTime"
type="bdt:DateTimeType"
                                        minOccurs="0">
                                        ...
                                </xsd:element>
                                <xsd:element name="EndDateTime"
type="bdt:DateTimeType"
                                        minOccurs="0">
                                        ...
                                </xsd:element>
                        </xsd:sequence>
                </xsd:choice>
        </xsd:sequence>
  </xsd:complexType>
```

2027 One technical identifier per aggregate kind (both shared and composite) will be used
2028 for both generic and structural referencing. It will be defined as an optional attribute
2029 named "**key**" to avoid any confusion with legacy XML ID attributes.

| | | |
|---|---|---|
| [R 8EA2] | Every aggregate business information entity (ABIE) **xsd:complexType** definition MUST contain an optional "**key**" attribute that MAY be used as the complex element identifier in a message instance. | 1 |
| [R 92C0] | The "**key**" attribute MUST be locally define on the ABIE **xsd:complexType** definition. "**key**" MUST be a reserved attribute name. | 1 |
| [R 8A37] | Every "**key**" local attribute MUST be of the type **xsd:token.** | 1 |

2030 ### 8.2.4 Element Declarations and References
2031 #### 8.2.4.1 ABIE Elements

2032 The content model of the ABIE complex type definitions will include both element
2033 declarations for BBIEs and ASBIEs. The BBIEs will always be declared locally. The
2034 rules for declaration of ASBIE's are exposed in the next section.

2035 Every ABIE must have a globally declared element. This global element reflects the
2036 unique DEN of the ABIE within the namespace to which it is assigned.

2037 **Note:**

2038 This rule applies even to ABIE's used in associations where the ASBIE
2039 AggregationKind is composition, resulting in a local element being used by the
2040 containing ABIE, as exposed in the next section.

| | | |
|---|---|---|
| [R 9DA0] | For each ABIE, a named **xsd:element** MUST be globally declared. | 1 |
| [R 9A25] | The name of the ABIE **xsd:element** MUST be the **ccts:DictionaryEntryName** with the separators and '**Details**' suffix removed and approved abbreviations and acronyms applied. | 1 |
| [R B27B] | Every ABIE global element declaration MUST be of the **xsd:complexType** that represents the ABIE. | 1 |
| [R 89A6] | For every attribute of an object class (BBIE) identified in an ABIE, a named **xsd:element** MUST be locally declared within the **xsd:complexType** representing that ABIE. | 1 |
| [R AEFE] | Each BBIE element name declaration MUST be the property term and qualifiers and the representation term of the basic business information entity (BBIE). | 1 |

| | | |
|---|---|---|
| [R 96D9] | Each BBIE element name declaration where the word '**identification**' is the final word of the property term and the representation term is '**identifier**', the term '**identification**' MUST be removed. | 1 |
| [R 9A40] | Each BBIE element name declaration where the word '**indication**' is the final word of the property term and the representation term is '**indicator**', the term '**indication**' MUST be removed from the property term. | 1 |
| [R A34A] | If the representation term of a BBIE is '**text**', '**text**' MUST be removed from the name of the element or type definition. | 1 |

#### 8.2.4.2 ASBIE Elements

The ASBIEs whose **ccts:AggregationKind** is Composition will always be declared locally.

| | | |
|---|---|---|
| [R 9025] | For every ASBIE whose **ccts:AggregationKind** is a composition, a named **xsd:element** MUST be locally declared. | 1 |
| [R A08A] | For each locally declared ASBIE, the element name MUST be the ASBIE property term and qualifier term(s) and the object class term and qualifier term(s) of the associated ABIE. | 1 |
| [R B27C] | For each locally declared ASBIE, the element declaration MUST use the **xsd:complexType** that represents its associated ABIE. | 1 |

For each ASBIE who's **ccts:AggregationKind** is not an AggregateKind composite, there are two mutually exclusive cases, one of which needs to be selected on the base of the applicable Message Assembly definition.

- The globally declared element for the associated ABIE is included in the content model of the parent ABIE as a nested complex property.
- An equivalent referencing element pointing to the associated ABIE is included in the content model of the parent ABIE.

| | | |
|---|---|---|
| [R 9241] | For every ASBIE whose AggregationKind is shared, where the association is implemented as a nested property, the globally declared element for the associated ABIE MUST be referenced using **xsd:ref**. | 1 |
| [R B78E] | Every ASBIE whose AggregationKind is not a composition, and where the association must be implemented as a referenced property, an equivalent referencing element pointing to the associated ABIE MUST be locally declared. | 1 |
| [R AEDD] | The equivalent referencing element MUST have a name composed of the ASBIE property term and property qualifier | 1 |

| | | |
|---|---|---|
| | term(s). | |
| [R B173] | For each equivalent referencing element a **xsd:complexType** MUST be declared. Its structure will be an empty element with a local attribute. | 1 |
| [R B523] | The name of the local attribute that is part of the empty element MUST be composed of the object class term and object qualifier term(s) of the ABIE being referenced, followed by the suffix '**Reference**'. | 1 |
| [R 8B0E] | The name of the **xsd:complexType** representing the equivalent referencing element MUST be composed of the object class term and object qualifier term(s) of the ABIE being referenced, followed by the suffix '**ReferenceType**'. | 1 |
| [R B7D6] | Each equivalent referencing element MUST be of the **xsd:complexType** that relates to the ABIE being referenced. | 1 |

Example 8-14 shows an ABIE type definition with a local element declaration for a BBIE ("ID"), a local element declaration for two ASBIEs ("SellerParty" and "BuyerParty") and a global element reference for the Invoice specific ABIE ("InvoiceTradeLineItem").

**Example 8-14: Element declaration and reference within an ABIE type definition**

```
<xsd:complexType name="InvoiceType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element name="SellerParty" type="ordman:SellerPartyType"/>
                <xsd:element name="BuyerParty" type="ordman:BuyerPartyType"/>
                <xsd:element ref="ordman:InvoiceTradeLineItem"
maxOccurs="unbounded"/>
         </xsd:sequence>
```

Example 8-15 shows the schema definition of an ASBIE specified as a referencing element, building on example 5.11.

**Example 8-15: Element and type definition of an ASBIE, specified as a referencing element**

```
  <xs:complexType name="PartyReferenceType">
         <xs:attribute name="partyReference" type="xs:token"/>
  </xs:complexType>

  <xs:element name="ClaimantParty" type="PartyReferenceType"/>
```

## 8.2.5 Annotation

### 8.2.5.1 Annotation Documentation

#### 8.2.5.1.1 ABIE Complex Type Definition

Every ABIE complexType definition must include a structured annotation documentation.

| | | |
|---|---|---|
| [R ACB9] | For every ABIE **xsd:complexType** definition a structured set of annotations MUST be present in the following pattern: | 1 |

| | | |
|---|---|---|
| | • UniqueID (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An identifier of the evolution over time of an ABIE instance.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be ABIE.<br>• DictionaryEntryName (mandatory): The official name of an ABIE.<br>• Definition (mandatory): The semantic meaning of an ABIE.<br>• ObjectClassName (mandatory): The Object Class Name of the ABIE.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object. | |

Example 8-16 shows the annotation documentation of an ABIE definition.

**Example 8-16: Annotation of an ABIE complexType Definition**

```
  <xsd:complexType name="AccountType" >
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
          <ccts:UniqueID>UNBE000000</ccts:UniqueID>
          <ccts:VersionID>0.00</ccts:VersionID>
          <ccts:CCTSArtifact>ABIE</ccts:CCTSArtifact>
          <ccts:DictionaryEntryName>String</ccts:DictionaryEntryName>
          <ccts:Definition>String</ccts:Definition>
          <ccts:ObjectClassName>String</ccts:ObjectClassName>
          <ccts:UsageRule>
                ...
          </ccts:UsageRule>
      </xsd:documentation>
    </xsd:annotation>
</xsd:complexType>
```

### 8.2.5.1.2 ABIE Element

Every ABIE element declaration must include structured annotation documentation.

| | | |
|---|---|---|
| [R 88B6] | For every ABIE **`xsd:element`** declaration definition, a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An identifier of the evolution over time of an ABIE instance.<br>• CCTSArtifact (mandatory): The abbreviation code of the type of component. In this case the value will always be ABIE.<br>• DictionaryEntryName (mandatory): The official name of an ABIE. | 1 |

| | | |
|---|---|---|
| | • Definition (mandatory): The semantic meaning of an ABIE.<br>• ObjectClassName (mandatory): The Object Class Name of the ABIE.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object. | |

2096 8.2.5.1.3 BBIE Element

2097 Every BBIE element declaration must include structured annotation documentation.

| | | |
|---|---|---|
| [R B8BE] | For every BBIE **`xsd:element`** declaration a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references a BBIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of a BBIE instance.<br>• SequencingKeyID (mandatory): Identifier of the sequence of the BBIE in the containing ABIE.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be BBIE.<br>• DictionaryEntryName (mandatory): The official name of the BBIE.<br>• Definition (mandatory): The semantic meaning of the BBIE.<br>• Cardinality (mandatory): Indication whether the BIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.<br>• ObjectClassQualifierName (optional): Qualifies the Object Class Name of the parent ABIE.<br>• ObjectClassName (mandatory): The Object Class Name of the parent ABIE.<br>• PropertyQualifierName (mandatory): Qualifies the Property Term of the BBIE.<br>• PropertyTermName (mandatory): The Property Term Name of the BBIE.<br>• RepresentationTermName (mandatory): Representation term.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the BBIE is commonly known and used in the business. | 1 |

| | | |
|---|---|---|
| | • Example (optional, repetitive): Example of a possible value of a BBIE. | |

Example 8-17 shows the annotation documentation of a BBIE Element.

**Example 8-17: Annotation of a BBIE Element**

```
<xsd:element name="ID" type="bdt:IDType" minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
          <ccts:UniqueID>UNBE000000</ccts:UniqueID>
          <ccts:VersionID>0.00</ccts:VersionID>
          <ccts:SequencingKeyID>1</ccts:SequencingKeyID>
          <ccts:CCTSArtifact>BBIE</ccts:CCTSArtifact>
          <ccts:DictionaryEntryName>String</ccts:DictionaryEntryName>
          <ccts:Definition>String</ccts:Definition>
          <ccts:Cardinality>String</ccts:Cardinality>
          <ccts:ObjectClassName>String</ccts:ObjectClassName>
          <ccts:PropertyTermName>String</ccts:PropertyTermName>
          <ccts:RepresentationTermName>String</ccts:RepresentationTermName>
    </xsd:documentation>
  </xsd:annotation>

</xsd:element>
```

#### 8.2.5.1.4 ASBIE Element

Every ASBIE element declaration must include structured annotation documentation.

| | | |
|---|---|---|
| [R 926A] | For every ASBIE **`xsd:element`** declaration a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references an ASBIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the ASBIE instance.<br>• SequencingKeyID (mandatory): Identifier of the sequence of the ASBIE in the containing ABIE.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be ASBIE.<br>• DictionaryEntryName (mandatory): The official name of the ASBIE.<br>• Definition (mandatory): The semantic meaning of the ASBIE.<br>• Cardinality (mandatory): Indication whether the ASBIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.<br>• ObjectClassQualifierName (optional): A term that qualifies the Object Class Name of the associating ABIE.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object. | 1 |

Example 8-18 shows the annotation documentation for an ASBIE element.

**Example 8-18: Annotation documentation definition for an ASBIE element**

```
<xsd:group name="ASBIEDocumentation">
        <xsd:sequence>
                <xsd:element name="UniqueID"
type="bdt:EntityUniqueIdentifierType"/>
                <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                <xsd:element name="SequencingKeyID"
type="bdt:SequencingKeyIdentifierType"/>
                <xsd:element name="CCTSArtifact"
type="bdt:DocumentationCCTSArtifactType" fixed="ASBIE"/>
                <xsd:element name="DictionaryEntryName" type="bdt:NameType"/>
                <xsd:element name="Definition" type="bdt:TextType"/>
                <xsd:element name="Cardinality" type="bdt:TextType"/>
                <xsd:element name="ObjectClassQualifierName" minOccurs="0"
maxOccurs="unbounded">
                        <xsd:complexType>
                                <xsd:complexContent>
                                        <xsd:extension base="bdt:TextType">
                                                <xsd:attribute name="orderKey"
type="xsd:positiveInteger" use="required"/>
                                        </xsd:extension>
                                </xsd:complexContent>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element name="UsageRule" type="ccts:UsageRuleType"/>
        </xsd:sequence>
  </xsd:group>
```

Example 8-19 shows a code snippet of the annotation documentation of an ASBIE Element.

**Example 8-19: Annotation of an ASBIE**

```
<xsd:element name="Status" type="ram:StatusType"  minOccurs="0"
maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
          <ccts:UniqueID>UNBE000000</ccts:UniqueID>
          <ccts:VersionID>0.00</ccts:VersionID>
          <ccts:SequencingKeyID>1</ccts:SequencingKeyID>
          <ccts:CCTSArtifact>ASBIE</ccts:CCTSArtifact>
          <ccts:DictionaryEntryName>String</ccts:DictionaryEntryName>
          <ccts:Definition>String</ccts:Definition>
          <ccts:Cardinality>String</ccts:Cardinality>
          <ccts:ObjectClassQualifierName>String</ccts:ObjectClassQualifierName>
          <ccts:UsageRule>
                ...
          </ccts:UsageRule>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

### 8.2.5.2 Annotation Application Information (AppInfo)

The annotation **`xsd:appInfo`** is expressed for all BIE artifacts defined in the BIE XML Schema files. The structure of the context is provided in section 7.5.2, Application Information (AppInfo). All contexts in which the BIE artifacts are applicable is expressed in the **`xsd:appInfo`**.

## 8.3 Business Data Type XML Schema Files

Ensuring consistency of business data types with the UN/CEFACT modularity and reuse goals requires creating a XML Schema file that defines business data types within the context category specified in the namespace. The business data type XML Schema file name must follow the UN/CEFACT XML Schema file naming approach. The business data type XML Schema file will contain the business data types which include the implementable core component data types for both unqualified and qualified data types. For this reason, the business data type XML Schema file will be used by the reusable BIE XML Schema file and all root XML Schema files defined within the same namespace.

### 8.3.1 Use of Business Data Type XML Schema Files

As defined in section 5.7.2, UN/CEFACT publishes a reference Business Data Type XML Schema that is comprised of XML Schema components representing the approved, unrestricted CCTS Business Data Type Catalogue BDT artifacts. Additional Business Data Type XML Schema is created to reflect both the unrestricted as well as restricted (qualified) BDTs that are used within a given namespace. These restrictions are implemented as an **`xsd:restriction`** or a new **`xsd:simpleType`**.

### 8.3.2 XML Schema Structure

Each business data type XML Schema file will be structured in a standard format to ensure consistency and ease of use.

The format is shown in Example 8-20 below and must adhere to the format of the relevant sections as detailed in Appendix B.

**Example 8-20: Structure of BDT XML Schema file**

```
<?xml version="1.0" encoding="utf-8"?>
<!-- ========================================================================= -->
<!-- =====  Business Data Type XML Schema File                          ===== -->
<!-- ========================================================================= -->
<!--
  Schema agency:         UN/CEFACT
    Schema version:      3.0
    Schema date:         15 July 2008



  Copyright (C) UN/CEFACT (2008). All Rights Reserved.

  ... see copyright information ...

-->
<xsd:schema targetNamespace=
  ... see namespace ...
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- ========================================================================= -->
  <!-- ===== Imports                                                       ===== -->
  <!-- ========================================================================= -->
  ... see imports ...
  <!-- ========================================================================= -->
  <!-- ===== Type Definitions                                              ===== -->
  <!-- ========================================================================= -->
  ... see type definitions ...
```

```
</xsd:schema>
```

### 8.3.3 Imports and Includes

The BDT XML Schema components are defined as **xsd:complexType** or **xsd:simpleType** directly within each BDT XML Schema file as necessary to support all Root XML Schema within a given namespace. Each BDT XML Schema file will use **xsd:include** for restricted code lists being used by BDT XML Schema Components within its parent namespace, and will also use **xsd:import** for any Common Code List XML Schema files being used by a BDT XML Schema Components within the BDT XML Schema's parent namespace.

Core Data Type XML Schema file is not directly imported or included, instead the data type are defined directly within the Business Data Type XML Schema file based on the requirements of the business implementation and the context category expressed in the namespace. The Restricted Code List XML Schema file will be defined within the context category expressed in the given namespace and included in the BDT XML Schema file.

The Common Code List XML Schema file is imported into the business data type XML Schema file so that the code list is used directly as defined by the code list definition.

| | | |
|---|---|---|
| [R 8E0D] | The BusinessDataType XML Schema file MUST include the RestrictedCodeList XML Schema files that are defined in the same namespace. | 1 |
| [R B4C0] | The BusinessDataType XML Schema file MUST import the CommonCodeList XML Schema files that it makes use of in the definition of the BDTs. | 1 |

### 8.3.4 Type Definitions

| | | |
|---|---|---|
| [R AE00] | Each CCTS BDT artifact within the UN/CEFACT Data Type Catalogue MUST be defined as an **xsd:simpleType** or **xsd:complexType**. | 1 |
| [R 973C] | The name of a business data type MUST be its dictionary entry name with separators and spaces removed. | 1 |

BDTs may have either their content or supplementary components restricted. Restricted BDT XML Schema Components are derived through restriction to the allowed **ccts:ContentComponent** facets **ccts:SupplementaryComponent** attributes of the unrestricted BDT type definition, unless non-standard variations from the base type are required. Non-standard variations will be defined as an **xsd:restriction** derivation from the unrestricted BDT TextType.

| | | |
|---|---|---|
| [R 80FD] | Every restricted Business Data Type XML Schema Component **xsd:type** definition MUST be derived from its base type using **xsd:restriction** unless a non-standard variation from the base | 1 |

| | | |
|---|---|---|
| | type is required. | |
| [R A9F6] | Every restricted Business Data Type XML Schema Component **`xsd:type`** definition requiring a non-standard variation from its base type MUST be derived from the BDT TextType XML Schema component. | 1 |

**Note:**

If a non-standard variation of the standard date time built-in data types is required, for example year month, then a qualified data type of the unqualified data type TextType needs to be defined, with the appropriate restriction specified, e.g. as a pattern, to specify the required format.

Example 8-21 shows examples of BDT definitions.

**Example 8-21: Type Definitions**

```
<!-- ========================================================================= -->
<!-- ===== Type Definitions                                             ===== -->
<!-- ========================================================================= -->
<!-- ===== Business Data Type based on DateTime Type                  ===== -->
<!-- ========================================================================= -->
<!-- =====  Day_ Date. Type                                             ===== -->
<!-- ========================================================================= -->
<xsd:simpleType name="DayDateType">
       <xsd:annotation>
              ... see annotation ...
       </xsd:annotation>
       <xsd:restriction base="xsd:gDay"/>
</xsd:simpleType>
...
<!-- ========================================================================= -->
<!-- =====  Description_ Text. Type                                     ===== -->
<!-- ========================================================================= -->
<xsd:complexType name="DescriptionTextType">
       <xsd:annotation>
              ... see annotation ...
       </xsd:annotation>
       <xsd:simpleContent>
              <xsd:restriction base="bdt:TextType"/>
       </xsd:simpleContent>
</xsd:complexType>
...
<!-- ========================================================================= -->
<!-- =====  Uniform Resource_ Identifier. Type                          ===== -->
<!-- ========================================================================= -->
<xsd:simpleType name="URIType">
       <xsd:annotation>
              ... see annotation ...
       </xsd:annotation>
       <xsd:restriction base="xsd:anyURI"/>
</xsd:simpleType>
...
<!-- ========================================================================= -->
<!-- =====  Country_ Identifier. Type                                   ===== -->
<!-- ========================================================================= -->
<xsd:simpleType name="CountryIDType">
       <xsd:annotation>
              ... see annotation ...
       </xsd:annotation>
       <xsd:restriction base="ids53166:CountryCodeContentType"/>
</xsd:simpleType>
...
```

| | | |
|---|---|---|
| [R AA60] | Every business data type based on a single codelist | 1 |

| | | |
|---|---|---|
| | **xsd:simpleType** MUST contain one of the following:<br>• **xsd:restriction** element with the **xsd:base** attribute set to the code lists defined simple type with appropriate namespace qualification or<br>• **xsd:union** element with, the **xsd:base** attribute set to the code list defined simple type and the **xsd:member** type attribute set to the code list defined simple types with appropriate namespace qualification. | |

XML Schema declarations for using code lists in business data types are shown in Example 8-22 through Example 8-25.

**Example 8-22: Usage of only one Code List**

```
  <xsd:simpleType name="TemperatureMeasureUnitCodeType">
        <xsd:annotation>
              ... see annotation ...
        </xsd:annotation>
        <xsd:restriction
base="clm6Recommendation20:MeasurementUnitCommonCodeContentType">
              <xsd:length value="3"/>
              <xsd:enumeration value="BTU">
                    <xsd:annotation>
                          <xsd:documentation xml:lang="en">
                                <ccts:Name>British thermal unit</ccts:Name>
                          </xsd:documentation>
                    </xsd:annotation>
              </xsd:enumeration>
              <xsd:enumeration value="CEL">
                    <xsd:annotation>
                          <xsd:documentation xml:lang="en">
                                <ccts:Name>degree Celsius</ccts:Name>
                          </xsd:documentation>
                    </xsd:annotation>
              </xsd:enumeration>
              <xsd:enumeration value="FAH">
                    <xsd:annotation>
                          <xsd:documentation xml:lang="en">
                                <ccts:Name>degree Fahrenheit</ccts:Name>
                          </xsd:documentation>
                    </xsd:annotation>
              </xsd:enumeration>
        </xsd:restriction>
  </xsd:simpleType>
```

**Example 8-23: Combination of Code Lists**

```
  <xsd:simpleType name="AccountDutyCodeType">
        <xsd:annotation>
              ... see annotation ...
        </xsd:annotation>
        <xsd:union memberType="clm64437:AccountTypeCodeContentType
              clm65153:DutyTaxFeeTypeCodeContentType"/>
  </xsd:simpleType>
```

**Example 8-24: Use of Choice for Alternative Code Lists**

```
  <xsd:complexType name="PersonPropertyCodeType">
        <xsd:annotation>
              ... see annotation ...
        </xsd:annotation>
        <xsd:choice>
              <xsd:element ref="clm63479:MaritalCode"/>
```

```
                <xsd:element ref="clm63499:GenderCode"/>
        </xsd:choice>
</xsd:complexType>
```

**Example 8-25: Use of Choice for Alternative Code Lists**

```
    <xsd:simpleType name="PersonPropertyCodeType">
            <xsd:union memberTypes="clm63479:MaritalCode
                        clm63499:GenderCode"/>
</xsd:simpleType>
```

| | | |
|---|---|---|
| [R AAD1] | Every business data type that has a choice of two or more code lists MUST be defined as one of the following:<br>• A **xsd:complexType** that contains the **xsd:choice** element whose content model consists of element references for the alternative code lists to be included with appropriate namespace qualification<br>• A **xsd:simpleType** that contains the **xsd:union** element whose **xsd:memberType** includes the simpleType definitions of the alternative code lists to be included with appropriate namespace qualification. | 1 |

### 8.3.5 Attribute and Element Declarations

There will be no element declarations in the BDT XML Schema files. There will be no global attribute declarations in the BDT XML Schema file. The only allowed attributes will be supplementary components.

| | | |
|---|---|---|
| [R 8B3D] | Global **xsd:element** declarations MUST NOT occur in the BDT XML Schema File. | 1 |
| [R B340] | Global **xsd:attribute** declarations MUST NOT occur in the BDT XML Schema File. | 1 |
| [R ACA7] | Local **xsd:attribute** declarations MUST only represent CCTS Supplementary Components for the Business Data Type for which they are being declared. | 1 |

### 8.3.6 Annotations

#### 8.3.6.1 Annotation Documentation

##### 8.3.6.1.1 BDT Types

Every BDT element and type declaration must include structured annotation documentation.

| | | |
|---|---|---|
| [R BFE5] | Every business data type definition MUST contain a structured set of annotation documentation in the following sequence and pattern: | 1 |

| | | |
|---|---|---|
| | • UniqueID (mandatory): The identifier that references a Business Data Type instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the Business Data Type instance.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be BDT.<br>• DictionaryEntryName (mandatory): The official name of the Business Data Type.<br>• Definition (mandatory): The semantic meaning of the Business Data Type.<br>• DataTypeQualifierName (mandatory): A name that qualifies the Representation Term in order to differentiate it from its underlying Core Data Type and other Business Data Type.<br>• DataTypeName (mandatory): Name of the DataType.<br>• PrimitiveTypeCode (mandatory): The primitive data type of the Business Data Type.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the BDT is commonly known and used in the business.<br>• Example (optional, repetitive): Example of a possible value of a Business Data Type. | |

Example 8-26 shows the annotation documentation for a BDT.

**Example 8-26: Annotation documentation definition for BDT**

```
<xsd:group name="BDTDocumentation">
         <xsd:sequence>
                <xsd:element name="UniqueID"
type="bdt:DataTypeUniqueIdentifierType"/>
                <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                <xsd:element name="CCTSArtifact"
type="bdt:DocumentationAcronymCodeType" fixed="BDT"/>
                <xsd:element name="DictionaryEntryName" type="bdt:NameType"/>
                <xsd:element name="Definition" type="bdt:TextType"/>
                <xsd:element name="DataTypeQualifierName" minOccurs="0"
maxOccurs="unbounded">
                        <xsd:complexType>
                                <xsd:complexContent>
                                        <xsd:extension base="bdt:TextType">
                                                <xsd:attribute name="orderKey"
type="xsd:positiveInteger" use="required"/>
                                        </xsd:extension>
                                </xsd:complexContent>
                        </xsd:complexType>
                </xsd:element>
                <xsd:element name="DataTypeName" type="bdt:NameType"/>
                <xsd:element name="PrimitiveTypeCode"
type="bdt:PrimitiveTypeCodeType" maxOccurs="unbounded"/>
                <xsd:element name="UsageRule" type="ccts:UsageRuleType"
minOccurs="0" maxOccurs="unbounded"/>
```

```
                <xsd:element name="BusinessTermName" minOccurs="0"
maxOccurs="unbounded"/>
                <xsd:element name="Example" type="bdt:TextType" minOccurs="0"
maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
```

Example 8-27 shows an example annotation documentation of a BDT.

**Example 8-27: Annotation of business data types**

```
... see type definition ...
<xsd:annotation>
  <xsd:documentation xml:lang="en-US">
          <ccts:UniqueID>UNDT000000-000</ccts:UniqueID>
          <ccts:VersionID>0.00</ccts:VersionID>
          <ccts:CCTSArtifact>BDT</ccts:CCTSArtifact>
          <ccts:DictionaryEntryName>String</ccts:DictionaryEntryName>
          <ccts:Definition>String</ccts:Definition>
          <ccts:DataTypeName>String</ccts:DataTypeName>
          <ccts:PrimitiveTypeCode>Binary</ccts:PrimitiveTypeCode>
          <ccts:UsageRule>
                ...
          </ccts:UsageRule>
  </xsd:documentation>
</xsd:annotation>
... see type definition ...
```

#### 8.3.6.1.2 BDT Type Supplementary Components

Every BDT Supplementary Component attribute declaration must include structured annotation documentation.

| | | |
|---|---|---|
| [R 9C95] | For every supplementary component **`xsd:attribute`** declaration a structured set of annotation documentations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references a Supplementary Component of a Core Component Type instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the BDT Supplementary Component instance.<br>• SequencingKeyID (mandatory): Identifier of the sequence of the BDT Supplementary Component.<br>• CCTSArtifact (mandatory): The type of component. In this case the value will always be BDTSC.<br>• DictionaryEntryName (mandatory): The official name of the ASBIE.<br>• Definition (mandatory): The semantic meaning of the ASBIE.<br>• DataTypeQualifierName (mandatory):<br>• DataTypeName (mandatory):<br>• PropertyTermName (mandatory): The Property Term Name of the associated Supplementary Component. | 1 |

| | | |
|---|---|---|
| | • RepresentationTermName (mandatory):<br>• PrimitiveTypeCode (mandatory):<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the BDT is commonly known and used in the business.<br>• Example (optional, repetitive): Example of a possible value of a Supplementary Component. | |

#### 8.3.6.2 Annotation Application Information (AppInfo)

The annotation **`xsd:appInfo`** is expressed for all BDT artifacts defined in the BDT XML Schema files. The structure of the context is provided in section 7.5.2, Application Information (AppInfo). All contexts in which the BDT artifacts are applicable is expressed in the **`xsd:appInfo`**.

## 8.4 Code List XML Schema Files

Codes are an integral component of any business to business information flow. Codes have been developed over time to facilitate the flow of compressed, standardized values that can be easily validated for correctness to ensure consistent data. In order for the XML instance documents to be fully validated by the parsers, any codes used within the XML document need to be available as part of the schema validation process. Many international, national and sectorial agencies create and maintain code lists relevant to their area. If required to be used within an information flow, these code lists will be stored in their own XML Schema file, and are referred to as external code lists. For example, many of the existing code lists that exist in the United Nations Code List (UNCL) will be stored as external code list XML Schema files for use within other UN/CEFACT XML Schema files.

| | | |
|---|---|---|
| [R 9E40] | Each UN/CEFACT maintained code list MUST be defined in its own XML Schema file. | 2 |

UN/CEFACT recognizes two basic types of code lists:

- Common code list are universally defined for all context which are generally maintained by standards bodies.
- Restricted code list which are defined as a subset or at times additions to existing common code lists. These code lists are defined within a given context of their use.

### 8.4.1 Shared Code List XML Schema Components

XML Schema Components that are the same for both Common Code List XML Schema Files and Restricted Code List XML Schema Files.

#### 8.4.1.1 Code List XML Schema Structure

Each Code List XML Schema file will be structured in a standardized format in order to ensure consistency and ease of use.

This structure is show in Example 8-28.

**Example 8-28: Structure of code lists**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ================================================================== -->
<!-- =====  6Recommendation20 - Code List XML Schema File             ===== -->
<!-- ================================================================== -->
<!--
   Schema agency:        UN/CEFACT
   Schema version:       2.0
   Schema date:          17 January 2006

   Code list name:       Measurement Unit Common Code
   Code list agency:     UNECE
   Code list version:    3

  Copyright (C) UN/CEFACT (2006). All Rights Reserved.

   ... see copyright information ...

-->
<xsd:schema targetNamespace=" ... see namespace ...
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- ================================================================== -->
  <!-- ===== Root Element                                          ===== -->
  <!-- ================================================================== -->
          ... see root element declaration ...
  <!-- ================================================================== -->
  <!-- ===== Type Definitions                                      ===== -->
  <!-- ================================================================== -->
  <!-- ===== Type Definition: Measurement Unit Common Code Content Type == -->
  <!-- ================================================================== -->
          ... see type definition ...
</xsd:schema>
```

#### 8.4.1.2 Code List XML Schema Name

The name of the code list schema files are dependent upon the agency that has defined them and the name of the code list it self.

| | | |
|---|---|---|
| [R 849E] | The XML Schema File, file name for code lists MUST be of the form:<br>**<Agency Identifier \| Agency Name Text>_<List Identification Identifier \| List Name Text>_<Version Identifier>.xsd**<br>All periods, spaces, or other separators are removed except for the "." before xsd and the "_" between the names.<br>Where:<br>• Agency Identifier = identifies the agency that manages the list. The default agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.<br>• Agency Name Text = the name of the agency that maintains | 2 |

| | | |
|---|---|---|
| | the list.<br>• List Identification Identifier = identifies a list of the respective corresponding codes or ids.<br>• List Name Text = the name of a list of codes.<br>• Version Identifier = identifies the version. | |

### 8.4.2 Common Code List XML Schema Components

Common code list that are universally defined for all contexts and maintained by standards bodies will be imported into the context specific namespaces that use them.

#### 8.4.2.1 Namespace Name for Common Code Lists

The namespace name for code list is somewhat unique in order to convey some of the supplementary component information rather than including them as attributes. Specifically, the UN/CEFACT namespace structure for a namespace name of a code list extends the earlier rules for namespace names.

| | | |
|---|---|---|
| [R 992A] | The XML Schema namespaces for code list XML Schema files MUST use the following pattern:<br>**URN:** `urn:<organization>:<org hierarchy> *[:<org hierarchy level n>]:codelist:common:<major>:<status>:<name>`<br>**URL:** `http://<organization>/<org hierarchy>*[/<org hierarchy level n>]/codelist/common/<major>/<status>/<name>`<br>Where:<br>• organization – Identifier of the organization providing the standard.<br>• org hierarchy – The first level of the hierarchy within the organization providing the standard.<br>• org hierarchy level – Zero to n level hierarchy of the organization providing the standard.<br>• codelist – A fixed value token for common codelists.<br>• common – A fixed value token for common codelists.<br>• major – The Major version number of the codelist.<br>• status – The status of the schema as: draft\|standard<br>• name – The name of the XML Schema file (using upper camel case) with periods, spaces, or other separators and the words 'schema module' removed. | 1 |

| | | |
|---|---|---|
| | ○ Code list names are further defined as: <Code List Agency Identifier\|Code List Agency Name Text> ><divider><Code List Identification Identifier\|Code List Name Text><br>▪ Where:<br>• Code List Agency Identifier – is the identifier for the agency that code list is from.<br>• Code List Agency Name Text – is the text of the name that the code list is from.<br>• Divider – the divider character for URN is ':' the divider character for URL is '/'.<br>• Code List Identification Identifer – is the identifier for the given code list.<br>• Code List Name Text – is the text of the name for the code list. | |

Example 8-29 shows a namespace name of a code list using an agency and a code list identifier at draft status.

**Example 8-29: Namespace name of a code list with an agency and a code list identifier at draft status**

```
"urn:un:unece:uncefact:codelist:common:D.04A:draft:6:3403: "
where
D.04A = the version of the UN/CEFACT directory
6 = the value for UN/ECE in UN/CEFACT data element 3055 representing
    the Code List. Agency. Identifier
3403 = UN/CEFACT data element tag for Name type code representing
    the Code List. Identification. Identifier
```

Example 8-30 shows a namespace name of a proprietary code list at draft status.

**Example 8-30: Namespace name of proprietary code list at draft status**

```
"urn:un:unece:uncefact:codelist:common:1:draft:Security_Initiative:Document_Securit
y"
where
SecurityInitiative = the code list agency name of a repsonsible agency, which
                     is not defined in UN/CEFACT data element 3055
                      representing the Code List. Agency. Identifier
DocumentSecurity = the value for Code List. Name. Text
1.2 = the value for Code List. Version. Identifier
```

Example 8-31 shows a namespace name of a code list with and agency and code list identifier at standard status.

**Example 8-31: Namespace name of a code list with an agency and a code list identifier at standard status**

```
"urn:un:unece:uncefact:codelist:common:D.04A:standard:6:3403"
where
6 = the value for UN/ECE in UN/CEFACT data element 3055 representing
    the Code List. Agency. Identifier
3403 = UN/CEFACT data element tag for Name status code representing
    the Code List. Identification. Identifier
D.04A = the version of the UN/CEFACT directory
```

Example 8-32 shows a namespace name of a proprietary code list at standard status.

**Example 8-32: Namespace name of proprietary code list at standard status**

```
"urn:un:unece:uncefact:codelist:common:1:standard:Security_Initiative:Document_Secu
rity"
where
SecurityInitiative = the code list agency name of a responsible agency, which
                     is not defined in UN/CEFACT data element 3055
                      representing the Code List. Agency. Identifier
DocumentSecurity = the value for Code List. Name. Text
1.2 = the value for Code List. Version. Identifier
```

While the versioning for code lists published by external organisations is outside of the UN/CEFACT control. UN/CEFACT published code lists in XML Schema files the value of the Code List Version Identifier will follow the rules for versioning other UN/CEFACT XML Schema files.

#### 8.4.2.2 XML Schema Namespace Token for Common Code Lists

A unique token will be defined for each namespace for common code lists. The token representing the namespace of common code lists should be constructed based on the identifier of the agency maintaining the code list and the identifier of the specific code list as issued by the maintenance agency except where there is no identifier. When there is no identifier, the name for the agency and/or code list should be used instead. This will typically be true when proprietary code lists are used. This method of token construction will provide uniqueness with a reasonably short token. When the code list is used for a business data type with a restricted set of valid code values, the business data type name is required to be used to distinguish one set of restricted values from another.

The agency maintaining the code list will generally be either identified by the agency code as specified in data element 3055 in the UN/CEFACT Code List directory or the agency name if the agency does not have a code value in 3055. The identifier of the specific code list will generally be the data element tag of the corresponding list in the UN/CEFACT directory. If there is no corresponding data element, then the name of the code list will be used.

In cases where the code list schema is a restricted set of values of a published code list schema, the code list schema will be associated with a business data type, and the name of the business data type will be included as part of the namespace token to ensure uniqueness from the unrestricted code list schema.

| | | |
|---|---|---|
| [R 9FD1] | Each UN/CEFACT maintained Common Code list XML Schema File MUST be represented by a unique token constructed as follows:<br>**clm[<Business data type name>]<Code List Agency Identifier\|Code List Agency Name Text><Code List Identification Identifier\|Code List Name Text>**<br>Where any repeated words are eliminated.<br>• Business Data Type Name – is the name of the business data type in the business data type XML Schema file. | 2 |

| | | |
|---|---|---|
| | • Code List Agency Identifier – is the identifier for the agency that code list is from.<br>• Code List Agency Name Text – is the text of the name that the code list is from.<br>• Code List Identification Identifier – is the identifier for the given code list.<br>• Code List Name Text – is the text of the name for the code list. | |

Example 8-33 shows a code list token with an agency and code list identifier.

**Example 8-33: Code list token with an agency and a code list identifier**

```
The code list token for Name Type. Code is clm63403
where
6 = the value for UN/ECE in UN/CEFACT data element 3055 representing
    the Code List. Agency. Identifier
3403 = UN/CEFACT data element tag for Name status code representing
    the Code List. Identification. Identifier
```

Example 8-34 shows a code list token for a business data type with an agency and code list identifiers.

**Example 8-34: Code list token for a qualified data type with an agency and code list identifiers**

```
Code list token for Person_Name Type. Code is clmPersonNameType63403
where
PersonNameType = name of the qualified data type
6 = the value for UN/ECE in UN/CEFACT data element 3055 representing
    the Code List. Agency. Identifier
3403 = UN/CEFACT data element tag for Name status code representing
    the Code List. Identification. Identifier
```

Example 8-35 shows a code list token for a proprietary code list.

**Example 8-35: Code list token for a proprietary code list**

```
Code list token for a proprietary code list for Document Security is
clmSecurityInitiativeDocumentSecurity
where
SecurityInitiative = the code list agency name of a repsonsible agency, which is
not defined in UN/CEFACT data element 3055
      representing the Code List. Agency. Identifier
DocumentSecurity = the value for Code List. Name. Text
```

Based on the constructs identified in the above examples, a namespace declaration for a code list would appear as shown in Example 8-36.

**Example 8-36: Target namespace declaration for a code list**

```
<xsd:schema
  targetNamespace="urn:un:unece:uncefact:codelist:common:D.04A:draft:6:4437"
  xmlns:clm64437=" urn:un:unece:uncefact:codelist:common:D.04A:draft:6:4437 "
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
```

2600 **Note:**
2601 External developers are encouraged to follow the above construct rule when
2602 customizing schema for code lists to ensure that there is no namespace conflict.

2603 #### 8.4.2.3 Imports and Includes

2604 UN/CEFACT Common Code List Schema Modules are standalone schema modules
2605 and will not import or include any other schema modules.

| | | |
|---|---|---|
| [R 86C8] | Common Code List XML Schema files MUST NOT import or include any other XML Schema Files. | 1 |

2606 #### 8.4.2.4 Type Definitions

| | | |
|---|---|---|
| [R A8EF] | In each Common Code List XML Schema File one, and only one, named **`xsd:simpleType`** MUST be defined for the content component. | 1 |
| [R 92DA] | In each Common Code List XML Schema File the name of the **`xsd:simpleType`** MUST be the name of code list root element with the word '**`ContentType`**' appended. | 1 |

2607 Example 8-37 shows a simple type definition used in a code list.

2608 **Example 8-37: Simple type definition of code lists**

2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620

```
<!-- ========================================================================= -->
<!-- ===== Type Definitions                                              ===== -->
<!-- ========================================================================= -->
<!-- =====  Type Definition: Account Type Code            ===== -->
<!-- ========================================================================= -->
<xsd:simpleType name="AccountTypeCodeContentType">
       <xsd:restriction base="xsd:token">
              <xsd:enumeration value="2">
                     ... see enumeration ...
              </xsd:enumeration>
       </xsd:restriction>
</xsd:simpleType>
```

2621 Each code list XML Schema file will have a single **`xsd:simpleType`** defined. This
2622 type definition will have a **`xsd:restriction`** expression whose base is a XML
2623 Schema built-in data type. The **`xsd:restriction`** will be used to convey the
2624 content component enumeration value(s).

| | | |
|---|---|---|
| [R B40B] | In each Common Code List XML Schema File the **`xsd:restriction`** element base attribute value MUST be set to **`xsd:token`**. | 1 |
| [R 962C] | Each code in a Common Code List MUST be expressed as an **`xsd:enumeration`**, where the **`xsd:value`** for the enumeration is the actual code value. | 1 |

2625 Example 8-38 shows an enumeration facet for a code list.

**Example 8-38: Enumeration facet of a code lists**

```
... see type defintion ...
  <xsd:enumeration value="2">
         <xsd:annotation>
                ... see annotation
         </xsd:annotation>
  </xsd:enumeration>
  <xsd:enumeration value="15">
         <xsd:annotation>
                ... see annotation
         </xsd:annotation>
  </xsd:enumeration>
  ...
```

#### 8.4.2.5 Element Declarations

| | | |
|---|---|---|
| [R 8D1D] | In each Common Code List XML Schema File a single root element MUST be globally declared within the given code list XML Schema file. | 1 |
| [R BE84] | In each Common Code List XML Schema File the code list root element MUST be of a type representing the actual list of code values represented by the type whose name ends in 'ContentType'. | 1 |

Example 8-39 shows a root element declaration for a code list.

**Example 8-39: Root element declaration of code lists**

```
<!-- ========================================================================= -->
<!-- ===== Root Element                                                  ===== -->
<!-- ========================================================================= -->
<xsd:element name="AccountTypeCode" type="clm64437:AccountTypeCodeContentType"/>
```

#### 8.4.2.6 Annotation

##### 8.4.2.6.1 Annotation Documentation

###### 8.4.2.6.1.1 Code List Documentation

Every Common Code List XML Schema file must include structured annotation documentation.

| | | |
|---|---|---|
| [R BFE5] | Every Common Code List MUST contain a structured set of annotation documentation in the following sequence and pattern:<br>• UniqueID (mandatory): The identifier that references a Business Data Type instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the Code List.<br>• Name (optional):<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be CLM. | 1 |

| | | |
|---|---|---|
| | • Description (mandatory):<br>• PrimitiveTypeCode (mandatory): The primitive data type of the Code List.<br>• ModificationAllowedIndicator (mandatory):<br>• DefaultIndicator (mandatory):<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the Code List is commonly known and used in the business. | |

Example 8-40 shows the declaration of the code list documentation structure.

**Example 8-40: Declaration of code lists documentation structure**

```
  <xsd:group name="CodeListDocumentation">
          <xsd:sequence>
                  <xsd:element name="UniqueID" type="bdt:IDType"/>
                  <xsd:element name="VersionID" type="bdt:IDType"/>
                  <xsd:element name="Name" type="bdt:NameType" minOccurs="0"/>
                  <xsd:element name="AgencyID" type="bdt:IDType"/>
                  <xsd:element name="AgencyName" type="bdt:NameType" minOccurs="0"/>
                  <xsd:element name="CCTSArtifact"
type="bdt:DocumentationCCTSARtifactCodeType"/>
                  <xsd:element name="Description" type="bdt:TextType"/>
                  <xsd:element name="PrimitiveTypeCode"
type="bdt:PrimitiveTypeCodeType"/>
                  <xsd:element name="ModificationAllowedIndicator"
type="bdt:IndicatorType" minOccurs="0"/>
                  <xsd:element name="DefaultIndicator" type="bdt:IndicatorType"
minOccurs="0"/>
                  <xsd:element name="UsageRule" type="ccts:UsageRuleType"
minOccurs="0" maxOccurs="unbounded"/>
                  <xsd:element name="BusinessTermName" minOccurs="0"
maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
```

#### 8.4.2.6.1.2 Code List Value Documentation

In order to facilitate a clear and unambiguous understanding of the list of allowable codes within an element, annotations will be provided for each enumeration to provide the code name and description.

| | | |
|---|---|---|
| [R A814] | Each code list **`xsd:enumeration`** MUST contain a structured set of annotations in the following sequence and pattern:<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be CLM<br>• Content (optional): The code of value for an enumeration.<br>• Name (optional): The name or text that the represents.<br>• Description (optional): Descriptive information concerning the code<br>• UsageRule (optional, repetitive): Indicates the Usage Rule | 1 |

| | | |
|---|---|---|
| | of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the Code List Value is commonly known and used in the business. | |

Example 8-41 shows the annotation documentation definition for the enumerations values of a code list.

**Example 8-41: Annotation documentation definition of the enumerations values of a code list**

```
<xsd:group name="CodeValueDocumentation">
          <xsd:sequence>
                    <xsd:element name="CCTSArtifact"
type="bdt:DocumentationCCTSArtifactType"/>
                    <xsd:element name="Content" type="xsd:string" minOccurs="0"/>
                    <xsd:element name="Name" type="bdt:NameType" minOccurs="0"/>
                    <xsd:element name="Description" type="bdt:TextType"/>
                    <xsd:element name="UsageRule" type="ccts:UsageRuleType"
minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element name="BusinessTermName" minOccurs="0"
maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
```

#### 8.4.2.6.2 Annotation Application Information (AppInfo)

Common Code List are intended to be applicable to all context as such they do not provide specific contexts.

### 8.4.3 Restricted Code List XML Schema Components

Users of the UN/CEFACT library may identify any subset or superset they wish from a specific code list for their own trading community by defining a business data type. This is accomplished through the use of Restricted Code List that do this for the context category expressed in the namespace.

Representation of a business data type of code lists could be

- A combination of several individual code lists using **`xsd:union`**
- A choice between several code lists, using **`xsd:choice`**
- Sub setting an existing code list using **`xsd:restriction`** or through defining the sub set directly.

Each of these can easily be accommodated in this syntax solution as required by the user's business requirements.

Restricted Code List are Code List XML Schema files that contain code lists that are applicable within the context category that is contained within the namespace that the restricted code list is defined. Restricted Code List XML Schema files contain a restricted subset of a code list.

A restricted code list XML Schema file maybe used where an existing common code list XML Schema file needs to be extended, where no suitable external code list XML Schema exists, or where the context in which the code list is to be used is well defined and expressed in the namespace.

| | | |
|---|---|---|
| [R 9FD1] | Restricted Code List XML Schema file MUST be used to<br>• Extend existing common code list or<br>• Define a codelist where one does not exist or<br>• Restrict the value of a common codelist for the context category in which it is defined. | 2 |

#### 8.4.3.1 Namespace Name for Restricted Code Lists

The namespace name for restricted code list uses the namespace for the context category in which it is defined. This is described earlier in this document.

#### 8.4.3.2 UN/CEFACT XML Schema Namespace Token for Restricted Code Lists

The namespace token for restricted code list uses the namespace token for the context category in which it is defined. This is described earlier in this document.

#### 8.4.3.3 Imports and Includes

Restricted Code List Schema Modules may import Common Code List XML Schema file if the Restricted Code List is restricting the Common Code List Schema file content.

| | | |
|---|---|---|
| [R 86C8] | Restrict Code List XML Schema files MUST NOT import or include any other XML Schema files other than possibly a Common Code List XML Schema file which it is restricting. | 1 |

# 9 XML Instance Documents

In order to be UN/CEFACT conformant, an instance document must be valid against the relevant UN/CEFACT compliant XML Schema file(s). The XML instance documents should be readable and understandable by both humans and applications, and should enable reasonably intuitive interactions. A XPath navigation path should describe the complete semantic understanding by concatenating the nested elements. This navigation path should also reflect the meaning of each dictionary entry name of a ABIE, BBIE or ASBIE.

This section further describes the requirements XML Instance documents:

- Character Encoding
- xsi:schemaLocation
- Empty Content
- xsi:type

## 9.1 Character Encoding

In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83) as agreed to by UN/CEFACT, all UN/CEFACT XML will be instantiated using UTF. UTF-8 is the preferred encoding, but UTF-16 may be used where necessary to support other languages.

| | | |
|---|---|---|
| [R ACE9] | All XML MUST be instantiated using UTF. UTF-8 should be used if possible, if not UTF-16 should be used. | 1 |

## 9.2 xsi:schemaLocation

The **`xsi:schemaLocation`** and **`xsi:noNamespaceLocation`** attributes are part of the XML schema instance namespace (http://www.w3.org/2001/XMLSchema-instance). To ensure consistency, the token **`xsi`** will be used to represent the XML schema instance namespace.

| | | |
|---|---|---|
| [R A1B9] | The **`xsi`** namespace prefix MUST be used to reference the "**`http://www.w3.org/2001/XMLSchema-instance`**" namespace and anything defined by the W3C XMLSchema-instance namespace. | 1 |

## 9.3 Empty Content

Empty elements do not provide the level of assurance necessary for business information exchanges and as such, will not be used.

The only case in which elements maybe empty are in cases of where the key and keyRef attributes are used to reference other entities in a given XML instance.

| | | |
|---|---|---|
| [R 9277] | The **`xsi:nil`** attribute MUST NOT appear in any conforming instance. | 1 |

## 9.4 xsi:type

The xsi:type attribute allows for substitution during an instantiation of a xml document. In the same way that substitution groups are not allowed, the xsi:type attribute is not allowed.

| | | |
|---|---|---|
| [R 8250] | The xsi:type attribute MUST NOT be used within an XML Instance. | 1 |

# 10 Use Cases for Common Code Lists

Code lists provide mechanisms for conveying data in a consistent fashion where all parties to the information – originator, sender, receiver, processor – fully understand the purpose, use, and meaning of the data. The UN/CEFACT XML NDRs support flexible use of code lists. This section details the mechanisms for such use.

The UN/CEFACT XML NDRs allow for five alternative uses for code lists:

- Referencing a predefined standard code list, such as ISO 4217 currency codes as a supplementary component in an business data type, such as bdt:AmountType.
- Referencing any code list, standard or proprietary, by providing the required identification as attributes in the business data type bdt:CodeType.
- Referencing a predefined code list by declaring a specific business data type.
- Choosing or combining values from several code lists.
- Restricting the set of allowed code values from an established code list.

Example 10-1 Code Use Example Schema is used as the basis for examples that illustrate how to implement each of these alternatives.

**Example 10-1: Code Use Example Schema**

```
<xsd:schema xmlns:ordman=":un:unece:cefact:data:ordermanagement:1:draft"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:un:unece:cefact:data:ordermanagement:1:draft"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<!-- =====  Include                                              ===== -->
  <xsd:include
schemaLocation="http://www.unece.org/uncefact/data/ordermanagement/1/draft/Business
InformationEntity_1p3p6.xsd"/>
  <xsd:include
schemaLocation="http://www.unece.org/uncefact/data/ordermanagement/1/draft/Business
DataType_1p3p6.xsd"/>


  <!-- Root element -->
  <xsd:element name="Invoice" type="ordman:InvoiceType"/>
  <!-- Messase type declaration -->
  <xsd:complexType name="InvoiceType">
         <xsd:sequence>
                <xsd:element name="Product" type="ordman:ProductType"/>
                <xsd:element name="CustomerParty" type="ordman:PartyType"/>
         </xsd:sequence>
  </xsd:complexType>
  <!-- The below type declaration would normally appear in a separate schema module
for all reusable components (ABIE) but is included here for completeness  -->
  <xsd:complexType name="ProductType">
         <xsd:sequence>
                <xsd:element name="TotalAmount" type="ordman:AmountType"/>
                <xsd:element name="TaxCurrencyCode" type="ordman:CodeType"/>
                <xsd:element name="ChangeCurrencyCode"
type="ordman:CurrencyCodeType"/>
                <xsd:element name="CalculationCurrencyCode"
type="ordman:CalculationCurrencyCodeType"/>
                <xsd:element name=”RestrictedCurrencyCode”
type=”ordman:RestrictedCurrencyCodeType”/>
         </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

This schema includes:

- The XML Schema file of all business data types defined for the given contect category (business process value which is order management), such as, AmountType, CodeType, QuantityType.
    - The two specific data types CurrencyCodeType and CalculationCurrencyCodeType are defined as restricted code list that are included through the BDT XML Schema File.
- The XML Schema file of all BIE defined for the given conext category such as **`PartyType`**.

Within the **`xsd:complexType`** of ProductType, five local elements are declared. Each of these elements represents one of the five different code list options.

## 10.1 Referencing a Common Code List in Business Data Types

In the Code Use Example Schema, the element TotalAmount is declared as shown in Example 10-2.

**Example 10-2: Declaration of TotalAmount Element**

```
<xsd:element name="TotalAmount" type="ordman:AmountType"/>
```

As shown in the element declaration, TotalAmount is of the CCTS business data type AmountType which has been defined in the UN/CEFACT business data type XML Schema file for the business process context category with the value of order management. The AmountType declaration is as show in Example 10-3.

**Example 10-3: Declaration of Amount DataTypes in the BDT**

```
<xsd:schema targetNamespace="urn:un:unece:uncefact:data:ordermanagement:1:draft"
xmlns:clm54217="urn:un:unece:uncefact:codelist:common:1:draft:5:4217:2001" …
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- ========================================================================= -->
  <!-- ===== Imports                                                       ===== -->
  <!-- ========================================================================= -->
  <!-- ===== Imports of Code Lists                                         ===== -->
  <!-- ========================================================================= -->
  <xsd:import namespace="urn:un:unece:uncefact:codelist:common:1:draft:5:4217:2001"
schemaLocation="
http://www.unece.org/uncefact/codelist/common/1/draft/5/4217_2001_.xsd "/>
  <!-- ========================================================================= -->
  <!-- ===== Type Definitions                                              ===== -->
  <!-- ========================================================================= -->
  <!-- ===== Amount Decimal. Type                                            =====
-->
  <!-- ========================================================================= -->
  <xsd:complexType name="AmountDecimalType">
         <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                       <xsd:attribute name="currencyCode"
type="clm5ISO42173A:ISO3AlphaCurrencyCodeContentType" use="optional"/>

                </xsd:extension>
         </xsd:simpleContent>
  </xsd:complexType>
```

This AmountType has attributes declared that represent the supplementary components defined in CCTS for this data type. These attributes include currencyCode for the supplementary component of Amount. Currency. Code. This currencyCode attribute is declared to be of the **`xsd:simpleType`**

**clm5ISO42173A:ISO3AlphaCurrencyCodeContentType**. The **clm5ISO42173A:ISO3AlphaCurrencyCodeContentType** has been declared in the code list schema module for ISO Currency Codes, and the allowed code values for the currencyCode attribute have been defined as enumeration facets in the **clm5ISO42173A:ISO3AlphaCurrencyCodeContentType** type definition.

An extract of the code list schema module for ISO Currency Codes is as shown in 10-4.

**Example 10-4: Declaration of a Currency Code List**

```
  <!-- ========================================================================= -->
  <!-- =====   Root Element Declarations                                   ===== -->
  <!-- ========================================================================= -->
  <xsd:element name="CurrencyCode" type="clm54217:CurrencyCodeContentType"/>
  <!-- ========================================================================= -->
  <!-- ===== Type Definitions                                              ===== -->
  <!-- ========================================================================= -->
  <!-- =====  Code List Type Definition: Currency Codes                    ===== -->
  <!-- ========================================================================= -->
  <xsd:simpleType name="CurrencyCodeContentType">
         <xsd:restriction base="xsd:token">
                <xsd:enumeration value="AED">
                       <xsd:annotation>
                              <xsd:documentation>
                                     <CodeName>Dirham</CodeName>
                              </xsd:documentation>
                       </xsd:annotation>
                </xsd:enumeration>
                <xsd:enumeration value="AFN">
                       <xsd:annotation>
                              <xsd:documentation>
                                     <CodeName>Afghani</CodeName>
                              </xsd:documentation>
                       </xsd:annotation>
                </xsd:enumeration>
         </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

The currencyCode attribute has a fixed value of ISO 4217 Currency Code as defined in CCTS. Thus, only code values from this code list are allowed in a CEFACT conformant instance document. In such an instance document, actual conveyance of a currency code value would be represented as:

```
         <TotalAmount currencyID="AED">3.14</TotalAmount>
```

It should be noted that when using this option, no information about the code list being used is carried in the instance document as this information is already defined in the underlying XML Schema.

### 10.1.1 Referencing any code list using BDT CodeType

The second element in our example message – TaxCurrencyCode – is of the business data type bdt:CodeType.

```
  <xsd:element name="TaxCurrencyCode" type="bdt:CodeType"/>
```

This **bdt:CodeType** data type includes a number of supplementary components required in order to uniquely identify the code list to be used for validation.

The **bdt:CodeType** is declared in the BDT XML Schema file shown in Figure 10-5

**Example 10-5: Declaration of a Code Type in the BDT XML Schema File**

```
  <xsd:complexType name="CodeType">
        <xsd:simpleContent>
              <xsd:extension base="xsd:token">
                    <xsd:attribute name="listID" type="xsd:token"
use="optional"/>
                    <xsd:attribute name="listName" type="xsd:string"
use="optional"/>
                    <xsd:attribute name="listAgencyID" type="xsd:token"
use="optional"/>
                    <xsd:attribute name="listAgencyName" type="xsd:string"
use="optional"/>
                    <xsd:attribute name="listVersionID" type="xsd:token"
use="optional"/>
                    <xsd:attribute name="listURI" type="xsd:anyURI"
use="optional"/>
              </xsd:extension>
        </xsd:simpleContent>
  </xsd:complexType>
```

When the **bdt:CodeType** is used, either the listURI (which will point uniquely to the code list) should be used, or a combination of the other attributes should be used. Thus, it is possible to refer to the code list relevant attributes either by the specific attributes for the explicit display of supplementary components, or by the list URI in which the value is based on the namespace name conventions.

The association to the specific namespace must be defined during runtime. In an instance document this element could be represented as:

```
<TaxCurrencyCode listName="ISO Currency Code" listAgencyName="ISO" listID="ISO
4217" listVersionID="2001" listAgencyID="5>AED</TaxCurrencyCode>
```

or

```
<TaxCurrencyCode
listURI="urn:un:unece:uncefact:codelist:draft:5:4217:2001">AED</TaxCurrencyCode>
```

It should be noted that when applying this option, validation of code values in the instance document will not be done by the XML parser.

### 10.1.2 Referencing a Common Code List in a BDT

The third element in our example message ChangeCurrencyCode is based on the business data type **bdt:CurrencyCodeType**.

```
              <xsd:element name="ChangeCurrencyCode"
type="bdt:CurrencyCodeType"/>
```

The **bdt:CurrencyCodeType** would be defined in the qualified data type schema module as:

```
  <xsd:simpleType name="CurrencyCodeType">
        <xsd:restriction base="clm54217-A:CurrencyCodeContentType"/>
  </xsd:simpleType>
```

This means that the value of the ChangeCurrencyCode element can only have code values from the identified ISO 4217 code list. In an instance document this element would be represented as:

```
<ChangeCurrencyCode>AED</ChangeCurrencyCode>
```

**Note:**

When using this option no information about the code list to be used is carried in the instance document as this is already defined in the XML schema.

## 10.2 Choosing or Combining Values from Several Code Lists

The fourth option is to chose or combine values from diverse code lists by using either the xsd:choice or xsd:union elements.

### 10.2.1 Choice

In the Code Use Example Schema, the element **CalculationCurrencyCode** is declared as:

```
<xsd:element name="CalculationCurrencyCode"
type="bdt:CalculationCurrencyCodeType"/>
```

The **CalculationCurrencyCode** element is business data type **bdt:CalculationCurrencyCodeType**.

The **bdt:CalculationCurrencyCodeType** is defined in the BDT XML Schema File as:

```
<xsd:complexType name="CalculationCurrencyCodeType">
       <xsd:choice>
              <xsd:element ref="clm54217-N:CurrencyCode"/>
              <xsd:element ref="clm54217-A:CurrencyCode"/>
       </xsd:choice>
</xsd:complexType>
```

The **xsd:choice** element provides a choice of values from either the **clm54217-N:CurrencyCode** or from **clm54217-A:CurrencyCode**. The schema module for **clm54217-A:CurrencyCode** is the same as the one used in section 10.1.1 above. The sample schema module for **clm54217-N:CurrencyCode** is shown in Example 10-6.

**Example 10-6: Sample clm54217-N:CurrencyCode Schema Module:**

```
<!-- ================================================================== -->
<!-- =====  Root Element Declarations                             ===== -->
<!-- ================================================================== -->
<xsd:element name="CurrencyCode" type="clm54217-N:CurrencyCodeContentType"/>
<!-- ===== Type Definitions                                       ===== -->
<!-- ================================================================== -->
<!-- =====  Code List Type Definition: 4217-N Currency Codes      ===== -->
<!-- ================================================================== -->
<xsd:simpleType name="CurrencyCodeContentType">
       <xsd:restriction base="xsd:token">
              <xsd:enumeration value="840">
```

```
                        <xsd:annotation>
                                <xsd:documentation>
                                        <CodeName>US Dollar</CodeName>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:enumeration>
                <xsd:enumeration value="978">
                        <xsd:annotation>
                                <xsd:documentation>
                                        <CodeName>Euro</CodeName>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:enumeration>
        </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

This **xsd:choice** option allows for the use of code values from different pre-defined code lists in the instance document. The specific code list being used in the instance document will be represented by the namespace prefix (**clm54217-A** or **clm54217-N**) being used for the namespace declaration of the imported code list and for the CurrencyCode element:

```
<Invoice … xmlns:clm54217-N="urn:un:unece:uncefact:codelist:draft:5:4217-N:2001" …
>
        <CalculationCurrencyCode>
                <clm54217-N:CurrencyCode>840</clm54217-N:CurrencyCode>
        </CalculationCurrencyCode>
        …
</Invoice>
```

The namespace prefix unambiguously identifies to the recipient of the instance from which code list each code value is defined.

### 10.2.2 Union

The **xsd:union** code list approach is similar to that for the **xsd:choice** approach in that multiple code lists are being used. The element declaration in the schema would be identical to that for choice in that the element **CalculationCurrencyCode** is still based on the business data type **bdt:CalculationCurrencyCodeType**.

```
  <xsd:element name="CalculationCurrencyCode"
type="qdt:CalculationCurrencyCodeType"/>
```

The difference is that the **bdt:CalculationCurrencyCodeType** would be defined in the BDT XML Schema File using an **xsd:union** element rather than an **xsd:choice** element:

```
    <xsd:simpleType name="CalculationCurrencyCodeType">
            <xsd:union memberTypes="clm54217-N:CurrencyCodeContentType
                    clm54217-A:CurrencyCodeContentType"/>
  </xsd:simpleType>
```

This declaration allowes the choice of values to come from either the **clm54217-N:CurrencyCodeContentType** or from the **clm54217-A:CurrencyCodeContentType**. The Common Code List XML Schema File for

**clm54217-A:CurrencyCodeContentType** is the same as the one used in Section 9.1.1 above. The Common Code List XML Schema File for **clm54217-N:CurrencyCodeContentType** is the same as the one used in Section 9.1.4.1.

This **xsd:union** option allows the use of code values from different pre-defined code lists in the instance document. The code lists must be imported once in the XML Schema File and must be shown once in the XML instance. The specific code list will be represented by the namespace prefix (**clm54217-A** or **clm54217-N**), but unlike the choice option, the element in the instance document will not have the specific code list token conveyed as the first part of the element name. The recipient of the instance does not know unambiguously which code list each code value is defined. This is because a reference to the specific code lists comes from different Code List XML Schema Files, such as, **clm54217-N** and **clm54217-A**.

In an instance document this element could be represented as:

```
<Invoice >
          …
          <CalculationCurrencyCode>840</CalculationCurrencyCode>
          …
</Invoice>
```

The advantage of the **xsd:union** approach is that attributes can make use of these code lists. For example, it may make sense for an implementation to standardize across the board on two currency code lists and have those apply to all of the data types, like **bdt:AmountType** and its currencyID attribute.

## 10.3 Restricting the Allowed Code Values

This option is used when it is desired to reduce the number of allowed code values from an existing code list. For example, a trading partner community may only recognize certain code values from the ISO 4217 Currency Code list. To accomplish this, create a Restricted Code List XML Schema File that contains the restricted set of value declarations in the namespace used for the context category that will use this Code List. This can be accomplished

- By importing the Common Code List XML Schema File and using **xsd:restriction** to restrict the values to the set of values required. Or
- By defining directly the set of value required as indicated in section *8.4.3 Restricted Code List XML Schema*

# Appendix A. Related Documents

The following documents provided significant levels of influence in the development of this document:

- UN/CEFACT Core Components Technical Specification Version 3.0 ODP 6 Implementation Verification
- UN/CEFACT Core Components Technical Specification, Part 8 of the ebXML Framework Version 2.01
- ebXML Technical Architecture Specification v1.04
- OASIS/ebXML Registry Information Model v2.0
- ebXML Requirements Specification v1.06
- Information Technology - Metadata registries: Framework for the Specification and Standardization of Data Elements, International Standardization Organization, ISO 11179-1
- Information Technology - Metadata registries: Classification of Concepts for the Identification of Domains, International Standardization Organization, ISO 11179-2
- Information Technology - Metadata registries: Registry Metamodel, International Standardization Organization, ISO 11179-3
- Information Technology - Metadata registries: Rules and Guidelines for the Formulation of Data Definitions, International Standardization Organization, ISO 11179-4
- Information Technology - Metadata registries: Naming and Identification Principles for Data Elements, International Standardization Organization, ISO 11179-5
- Information Technology - Metadata registries: Framework for the Specification and Standardization of Data Elements, International Standardization Organization, ISO 11179-6

# Appendix B. Overall Structure

The structure of an UN/CEFACT compliant XML schema must contain one or more of the following sections as relevant. Relevant sections must appear in the order given:

- XML Declaration
- Schema Module Identification and Copyright Information
- Schema Start-Tag
- Includes
- Imports
- Element
- Root Element
- Global Elements
- Type Definitions

## B.1 XML Declaration

A UTF-8 encoding is adopted throughout all UN/CEFACT XML schema.

**Example B-1: XML Declaration**

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

## B.2 Schema Module Identification and Copyright Information

**Example B-2: Schema Module Identification and Copyright Information**

```
<!-- ========================================================================= -->
<!-- =====  Example - Schema Module Name                                  ===== -->
<!-- ========================================================================= -->
<!--
   Schema agency:              UN/CEFACT
   Schema version:             3.0
   Schema date:                03 August 2008




  Copyright (C) UN/CEFACT (2006). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and
derivative works that comment on or otherwise explain it or assist in its
implementation may be prepared, copied, published and distributed, in whole or in
part, without restriction of any kind, provided that the above copyright notice and
this paragraph are included on all such copies and derivative works. However, this
document itself may not be modified in any way, such as by removing the copyright
notice or references to UN/CEFACT, except as needed for the purpose of developing
UN/CEFACT specifications, in which case the procedures for copyrights defined in
the UN/CEFACT Intellectual Property Rights document must be followed, or as
required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by
UN/CEFACT or its successors or assigns.
```

```
This document and the information contained herein is provided on an "AS IS" basis
and UN/CEFACT DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT
LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE.
-->
```

## B.3 Schema Start-Tag

The Schema Start-Tag section of an UN/CEFACT compliant XML schema must contain one or more of the below declarations as relevant. Relevant declarations must appear in the order given:

- Version
- Namespaces
- targetNamespace attribute
- xmlns:xsd attribute
- namespace declaration for current schema
- namespace declaration for reusable ABIEs actually used in the schema
- namespace declaration for unqualified data types actually used in the schema
- namespace declaration for qualified data types actually used in the schema
- namespace declaration for code lists actually used in the schema
- namespace declaration for identifier schemes actually used in the schema
- namespace declaration for CCTS
- Form Defaults
- elementFormDefault
- attributeFormDefault
- Others
- other schema attributes with schema namespace
- other schema attributes with non-schema namespace

**Example B-3: XML Schema Start Tag**

```
<xsd:schema
targetNamespace="urn:un:unece:uncefact:data:common:1:draft:Examples"
xmlns:rsm="urn:un:unece:uncefact:data:common:1:draft:Examples"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:com="urn:un:unece:uncefact:data:common:1:draft:"
xmlns:ids53166="urn:un:unece:uncefact:codelist:common:1997:draft:5:3166-1:1997"
xmlns:ids53166-2="urn:un:unece:uncefact:codelist:common:1998:draft:5:3166-2:1998"
xmlns:clm65153="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:5153:D.01C"
xmlns:clm64405="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:4405:D.01C "
xmlns:clm69143="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:9143:D.01C "
xmlns:clmPerson_Characteristic_Code63289="urn:un:unece:uncefact:codelist:common:D.0
1C:draft: 6:3289:D.01C"
xmlns:clm63479="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:3479:D.01C"
xmlns:clm63499="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:3499:D.01C"
xmlns:clm1161131="urn:un:unece:uncefact:codelist:common:4031:draft:11:61131:4031"
xmlns: clm66411="urn:un:unece:uncefact:codelist:common:2001:draft:6:6411:2001"
```

```
xmlns:clm54217="urn:un:unece:uncefact:codelist:common:2001:draft:5:4217:2001"
xmlns:clm5639="urn:un:unece:uncefact:codelist:common:1988Ldraft:5:639:1988"
xmlns:clm64437="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:4437:D.01C"

elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

## B.4 Includes

The Include section of an UN/CEFACT compliant XML schema must contain one or more of the below declarations as relevant. Relevant declarations must appear in the order given:

- Inclusion of the context category specific BIE XML Schema file.
- Inclusion of the context category specific BDT XML Schema file.
- Inclusion of the context category specific Restricted Code List XML Schema Files if used

**Example B-4: Includes**

```
<!-- ========================================================== -->
<!-- =====  Include                                         ===== -->
<!-- ========================================================== -->
<!-- =====  Inclusion of context category BIE XML Schema File  ===== -->
<!-- ========================================================== -->
  <xsd:include
schemaLocation="http://www.unece.org/uncefact/data/common/1/draft/BusinessInformati
onEntity_1p3p6.xsd"/>
<!-- ========================================================== -->
<!-- =====  Inclusion of context category BDT XML Schema File  ===== -->
<!-- ========================================================== -->
  <xsd:include
schemaLocation="http://www.unece.org/uncefact/data/common/1/draft/BusinessDataType_
1p3p6.xsd"/>
<!-- ========================================================== -->
<!-- =====  Inclusion of context category Code List XML Schema File ===== -->
<!-- ========================================================== -->
  <xsd:include
schemaLocation="http://www.unece.org/uncefact/data/common/1/draft/CodeList_1p3p6.xs
d"/>
```

## B.5 Imports

The Import section of an UN/CEFACT compliant XML Schema File must contain one or more of the below declarations as relevant. Relevant declarations must appear in the order given:

- Import of Common Code List XML Schema Files actually used

**Example B-5: Imports**

```
<!-- ========================================================== -->
<!-- =====  Import of Code lists                            ===== -->
<!-- ========================================================== -->
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:4437:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/comon/D.01C/draft/64437_D.01
C.xsd"/>
```

```
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:2001:draft:6:6411:2001"
schemaLocation="
http://www.unece.org/uncefact/codelist/common/2001/draft/66411_2001.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:2001:draft:5:4217:2001"
schemaLocation="
http://www.unece.org/uncefact/codelist/common:2001:draft/54217_2001.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:common:1988:draft:5:639-
1:1988"
schemaLocation="http://www.unece.org/uncefact/codelist/common/1998/draft/5639-
1.1988.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:4031:draft:11:61131:4031"
schemaLocation="http://www.unece.org/uncefact/codelist/common/4031/draft/1161131_40
31.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:3499:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/common/D.01C/draft/63499_D.0
1C.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:3479:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/common/D.01C/draft/63479_D.0
1C.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:3289:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/common/D.01C/draft/63289_D.0
1C.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:9143:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/common/D.01C/draft/69143_D.0
1C.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:4405:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/common/D.01C/draft/64405_D.0
1C.xsd"/>
<xsd:import
namespace="urn:un:unece:uncefact:codelist:common:D.01C:draft:6:5153:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/common/D.01C/draft/65153_D.0
1C.xsd"/>
```

## B.6 Elements

The root element is declared first when needed in schema that are used to support instance documents. Global elements are then declared following the root element when it is present.

**Example B-6:**

```
<!-- ============================================================== -->
<!-- =====  Element Declarations                                ===== -->
<!-- ============================================================== -->
<!-- =====  Root element                                        ===== -->
<!-- ============================================================== -->
  <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]>
<!-- ============================================================== -->
<!-- =====  Global Element Declarations                         ===== -->
<!-- ============================================================== -->
  <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]>
<!-- ============================================================== -->
```

## B.7 Root element

The root element's type definition is defined immediately following the definition of the global root element to provide clear visibility of the root element's type, of which this particular schema is all about.

**Example B-7:**

```
<!-- ================================================================== -->
<!-- =====  Root element                                          ===== -->
<!-- ================================================================== -->
  <xsd:element name="Invoice" type="rsm:InvoiceType">
         <xsd:annotation>
                <xsd:documentation>
                       <ccts:UniqueID>UNM0000001</ccts:UniqueID>
                       <ccts:Acronym>RSM</ccts:Acronym>
                       <ccts:Name>Invoice</ccts:Name>
                       <ccts:Version>1.0</ccts:Version>
                       <ccts:Description>A document that contains information
directly relating to
                              the economic event of ordering
products.</ccts:Description>
                       <ccts:BusinessProcessContextValue>Purchase
Order</ccts:BusinessProcessContextValue>
                </xsd:documentation>
         </xsd:annotation>
  </xsd:element>
```

**Example B-8: Global elements**

```
<!-- ================================================================== -->
<!-- ===== Global element                                       ===== -->
<!-- ================================================================== -->
 <xsd:element name="BuyerParty" type="ram:BuyerPartyType"/>
         <xsd:annotation>
                <xsd:documentation>
         <ccts:UniqueID>UNM0000002</ccts:UniqueID>
         <ccts:Acronym>RAM</ccts:Acronym>
         <ccts:DictionaryEntryName>Buyer_Party. Details</ccts:DictionaryEntryName>
         <ccts:Version>1.0</ccts:Version>
         <ccts:Definition>The party that buys.</ccts:Definition>
         <ccts:ObjectClassTerm>Party<ccts:ObjectClassTerm>
         <ccts:QualifierTerm>Buyer<ccts:QualifierTerm>
       </xsd:documentation>          </xsd:annotation>
  </xsd:element>
```

## B.8 Type Definitions

The definition of the BIEs used within the specific XML Schema File or by the XML Schema Files that make use of a common XML Schema File.

- Definition of types for Basic Business Information Entities in alphabetical order, if applicable.
- Definition of types for Aggregate Business Information Entities in alphabetical order, if applicable.

**Example B-9: Type Definitions**

```
<!-- ================================================================== -->
<!-- =====  Type Definitions                                      ===== -->
<!-- ================================================================== -->
<!-- =====  Type Definition: Account type                         ===== -->
```

```
<!-- ================================================================ -->
  <xsd:complexType name="AccountType">
         <xsd:annotation>
                <xsd:documentation xml:lang="en">
                       <ccts:UniqueID>UN00000001</ccts:UniqueID>
                       <ccts:Acronym>ABIE</ccts:Acronym>
                       <ccts:DictionaryEntryName>Account.
Details</ccts:DictionaryEntryName>
                       <ccts:Version>1.0</ccts:Version>
                        <ccts:Definition>A business arrangement whereby debits and/or
credits arising from transactions are recorded. This could be with a bank, i.e. a
financial account, or a trading partner offering supplies or services 'on account',
i.e. a commercial account</ccts:Definition>
         <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>
                </xsd:documentation>
         </xsd:annotation>
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType" minOccurs="0"
maxOccurs="unbounded">
                       <xsd:annotation>
                              <xsd:documentation xml:lang="en">
                                     <ccts:UniqueID>UN00000002</ccts:UniqueID>
                                     <ccts:Acronym>BBIE</ccts:Acronym>
                                     <ccts:DictionaryEntryName>Account.
Identifier</ccts:DictionaryEntryName>
                                     <ccts:Version>1.0</ccts:Version>
                                     <ccts:Definition>The identification of a
specific account.</ccts:Definition>
                                     <ccts:Cardinality>0..n</ccts:Cardinality>

  <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>

  <ccts:PropertyTerm>Identifier</ccts:PropertyTerm>

  <ccts:PrimaryRepresentationTerm>Identifier</ccts:PrimaryRepresentationTerm>
                                     <ccts:BusinessTerm>Account
Number</ccts:BusinessTerm>
                              </xsd:documentation>
                       </xsd:annotation>
                </xsd:element>
                <xsd:element name="Status" type="ram:StatusType" minOccurs="0"
maxOccurs="unbounded">
                       <xsd:annotation>
                              <xsd:documentation xml:lang="en">
                                     <ccts:UniqueID>UN00000003</ccts:UniqueID>
                                     <ccts:Acronym>ASBIE</ccts:Acronym>
                                     <ccts:DictionaryEntryName>Account.
Status</ccts:DictionaryEntryName>
                                     <ccts:Version>1.0</ccts:Version>
                                     <ccts:Definition>Status information related
to account details.</ccts:Definition>
                                     <ccts:Cardinality>0..n</ccts:Cardinality>


  <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>

  <ccts:PropertyTerm>Status</ccts:PropertyTerm>

  <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>
                                     <ccts:AssociatedObjectClassTerm>Status
                                            </ccts:AssociatedObjectClassTerm>

  <ccts:AssociationType>Aggregate</ccts:AssociationType>
                              </xsd:documentation>
                       </xsd:annotation>
                </xsd:element>
                <xsd:element name="Name" type="bdt:NameType" minOccurs="0"
maxOccurs="unbounded">
                       <xsd:annotation>
                              <xsd:documentation xml:lang="en">
                                     <ccts:UniqueID>UN00000004</ccts:UniqueID>
                                     <ccts:Acronym>BBIE</ccts:Acronym>
                                     <ccts:DictionaryEntryName>Account. Name.
Text</ccts:DictionaryEntryName>
                                     <ccts:Version>1.0</ccts:Version>
```

```
                                        <ccts:Definition>The text name for a
specific account</ccts:Definition>
                                        <ccts:Cardinality>0..n</ccts:Cardinality>


   <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>
                                        <ccts:PropertyTerm>Name</ccts:PropertyTerm>

   <ccts:PrimaryRepresentationTerm>Text</ccts:PrimaryRepresentationTerm>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="CurrencyCode" type="qdt:CurrencyCodeType"
minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                <xsd:documentation xml:lang="en">
                                        <ccts:UniqueID>UN00000005</ccts:UniqueID>
                                        <ccts:Acronym>BBIE</ccts:Acronym>
                                        <ccts:DictionaryEntryName>Account.
Currency. Code</ccts:DictionaryEntryName>
                                        <ccts:Version>1.0</ccts:Version>
                                        <ccts:Definition>A code specifying the
currency in which monies are held within the account.</ccts:Definition>
                                        <ccts:Cardinality>0..n</ccts:Cardinality>

   <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>

   <ccts:PropertyTerm>Currency</ccts:PropertyTerm>

   <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="TypeCode" type="qdt:AccountTypeCodeType"
minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                <xsd:documentation xml:lang="en">
                                        <ccts:UniqueID>UN00000006</ccts:UniqueID>
                                        <ccts:Acronym>BBIE</ccts:Acronym>
                                        <ccts:DictionaryEntryName>Account. Type.
Code</ccts:DictionaryEntryName>
                                        <ccts:Version>1.0</ccts:Version>
                                        <ccts:Definition>This provides the ability
to indicate what type of account this is (checking, savings,
etc).</ccts:Definition>
                                        <ccts:Cardinality>0..1<ccts:Cardinality>

   <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>
                                        <ccts:PropertyTerm>Type</ccts:PropertyTerm>

   <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Country" type="ram:CountryType" minOccurs="0"
maxOccurs="unbounded">
                        <xsd:annotation>
                                <xsd:documentation xml:lang="en">
                                        <ccts:UniqueID>UN00000007</ccts:UniqueID>
                                        <ccts:Acronym>ASBIE</ccts:Acronym>
                                        <ccts:DictionaryEntryName>Account.
Country</ccts:DictionaryEntryName>
                                        <ccts:Version>1.0</ccts:Version>
                                        <ccts:Definition>Country information
related to account details.</ccts:Definition>
                                        <ccts:Cardinality>0..n<ccts:Cardinality>

   <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>

   <ccts:PropertyTerm>Country</ccts:PropertyTerm>
                                        <ccts:AssociatedObjectClassTerm>Country
                                                </ccts:AssociatedObjectClassTerm>

   <ccts:AssociationType>Aggregate</ccts:AssociationType>
                                </xsd:documentation>
                        </xsd:annotation>
```

```
                </xsd:element>
                <xsd:element name="Person" type="ram:PersonType" minOccurs="0"
maxOccurs="unbounded">
                        <xsd:annotation>
                                <xsd:documentation xml:lang="en">
                                        <ccts:UniqueID>UN00000008</ccts:UniqueID>
                                        <ccts:Acronym>ASBIE</ccts:Acronym>
                                        <ccts:DictionaryEntryName>Account.
Person</ccts:DictionaryEntryName>
                                        <ccts:Version>1.0</ccts:Version>
                                        <ccts:Definition>Associated person
information related to account details.  This can be used to identify multiple
people related to an account, for instance, the account holder.</ccts:Definition>
                                        <ccts:Cardinality>0..n<ccts:Cardinality>

  <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>

  <ccts:PropertyTerm>Person</ccts:PropertyTerm>
                                        <ccts:AssociatedObjectClassTerm>Person
                                                </ccts:AssociatedObjectClassTerm>

  <ccts:AssociationType>Aggregate</ccts:AssociationType>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="Organisation" type="ram:OrganisationType"
minOccurs="0" maxOccurs="unbounded">
                        <xsd:annotation>
                                <xsd:documentation xml:lang="en">
                                        <ccts:UniqueID>UN00000009</ccts:UniqueID>
                                        <ccts:Acronym>ASBIE</ccts:Acronym>
                                        <ccts:DictionaryEntryName>Account.
Organisation</ccts:DictionaryEntryName>
                                        <ccts:Version>1.0</ccts:Version>
                                        <ccts:Definition>The associated
organisation information related to account details.  This can be used to identify
multiple organisations related to this account, for instance, the account
holder.</ccts:Definition>
                                        <ccts:Cardinality>0..n<ccts:Cardinality>

  <ccts:ObjectClassTerm>Account</ccts:ObjectClassTerm>

  <ccts:PropertyTerm>Organisation</ccts:PropertyTerm>
                <ccts:AssociatedObjectClassTerm>Organisation
                  </ccts:AssociatedObjectClassTerm>

  <ccts:AssociationType>Composition</ccts:AssociationType>
                                </xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
        </xsd:sequence>
  </xsd:complexType>
```

**Example B-10: Complete Structure**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ================================================================== -->
<!-- =====  [SCHEMA MODULE TYPE] Schema Module                    ===== -->
<!-- ================================================================== -->
<!--
  Schema agency:                  [SCHEMA AGENCY NAME]
    Schema version:             [SCHEMA VERSION]
    Schema date:                [DATE OF SCHEMA]

    [Code list name:]           [NAME OF CODE LIST]
    [Code list agency:]         [CODE LIST AGENCY]
    [Code list version:]        [VERSION OF CODE LIST]
    [Identifier list name:]     [NAME OF IDENTIFIER LIST]
    [Identifier list agency:]   [IDENTIFIER LIST AGENCY]
    [Identifier list version:]  [VERSION OF IDENTIFIER LIST]


  Copyright (C) UN/CEFACT (2006). All Rights Reserved.
```

```
This document and translations of it may be copied and furnished to others, and
derivative works that comment on or otherwise explain it or assist in its
implementation may be prepared, copied, published and distributed, in whole or in
part, without restriction of any kind, provided that the above copyright notice and
this paragraph are included on all such copies and derivative works. However, this
document itself may not be modified in any way, such as by removing the copyright
notice or references to UN/CEFACT, except as needed for the purpose of developing
UN/CEFACT specifications, in which case the procedures for copyrights defined in
the UN/CEFACT Intellectual Property Rights document must be followed, or as
  required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by
UN/CEFACT or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis
and UN/CEFACT DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT
LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL  NOT INFRINGE
ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
PURPOSE.
-->
<xsd:schema
targetNamespace="urn:un:unece:uncefact:data:draft:[MODULENAME]:[VERSION"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
… FURTHER NAMESPACES ….
elementFormDefault="qualified" attributeFormDefault="unqualified">
<!-- ========================================================================= -->
<!-- =====  Include                                                      ===== -->
<!-- ========================================================================= -->
<!-- =====  Inclusion of [TYPE OF MODULE]                                ===== -->
<!-- ========================================================================= -->
  <xsd:include schemaLocation="…"/>
<!-- ========================================================================= -->
<!-- =====  Imports                                                      ===== -->
<!-- ========================================================================= -->
<!-- =====  Import of [TYPE OF MODULE]                                   ===== -->
<!-- ========================================================================= -->
<xsd:import namespace="…" schemaLocation="…"/>
<!-- ========================================================================= -->
<!-- =====  Element Declarations                                         ===== -->
<!-- ========================================================================= -->
<!-- =====  Root element                                                 ===== -->
<!-- ========================================================================= -->
  <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]>
<!-- ========================================================================= -->
<!-- =====  Global Element Declarations                                  ===== -->
<!-- ========================================================================= -->
  <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]>
<!-- ========================================================================= -->
<!-- =====  Type Definitions                                             ===== -->
<!-- ========================================================================= -->
<!-- =====  Type Definition: [TYPE]                                      ===== -->
<!-- ========================================================================= -->
  <xsd:complexType name="[TYPENAME]">
         <xsd:restriction base="xsd:token">
                ... see type definition ....
         </xsd:restriction>
  </xsd:complexType>
</xsd:schema>
```

# Appendix C. ATG Approved Acronyms and Abbreviations

The following constitutes a list of ATG approved acronyms and abbreviations which must be used within tag names when these words are part of the dictionary entry name:

ABIE – Aggregate Business Information Entity

ACC – Aggregate Core Components

BBIE – Basic Business Information Entity

BCC – Basic Core Component

BDT – Business Data Type

BIE – Business Information Entity

CC – Core Components

ID – Identifier

URI – Uniform Resource Identifier

URL – Uniform Resource Locators

URN – Uniform Resource Name

UUID – Universally Unique Identifier

# Appendix D. Core Component XML Schema File

The Core Component XML Schema File is published as a separate file – CoreComponentType_3p0.xsd.

# Appendix E. Business Data Type XML Schema File

The Business Data Type XML Schema File is published as a separate file – BusinessDataType_3p0.xsd.

# Appendix F. Annotation Templates

# Appendix G. Mapping of CCTS Representation Terms to CCT and BDT Data Types

The following table represents the mapping between the representation terms as defined in CCTS and their equivalent data types as declared in the CCT schema module and the BDT schema module.

| Representation Term | Data Type for CCT | Data Type for BDT |
|---|---|---|
| Amount | xsd:decimal | xsd:decimal |
| | | |
| Binary Object | xsd:base64Binary | xsd:base64Binary |
| Graphic | | xsd:base64Binary |
| Sound | | xsd:base64Binary |
| Video | | xsd:base64Binary |
| | | |
| Code | xsd:token | xsd:token |
| | | |
| Date Time | xsd:string | xsd:dateTime |
| Date | | xsd:date |
| Time | | xsd:time |
| | | |
| Identifier | xsd:token | xsd:token |
| | | |
| Indicator | xsd:string | xsd:boolean |
| | | |
| Measure | xsd:decimal | xsd:decimal |
| Value | | xsd:decimal |
| Percent | | xsd:decimal |

| | | |
|---|---|---|
| Rate | | xsd:decimal |
| | | |
| Numeric | xsd:string | xsd:decimal |
| | | |
| Quantity | xsd:decimal | xsd:decimal |
| | | |
| Text | xsd:string | xsd:string |
| Name | | xsd:string |

3669

# Appendix H. Naming and Design Rules List

<table>
<tr><th>Rule ID</th><th>Rule Text</th><th>Categorization</th></tr>
<tr><td>[R B998]</td><td>Conformance SHALL be determined through adherence to the content of the normative sections and rules. Furthermore each rule is categorized to indicate the intended audience for the rule by the following:
<table>
<tr><th colspan="2">Rule Categorization</th></tr>
<tr><th>ID</th><th>Description</th></tr>
<tr><td>1</td><td>Rules which must not be violated by individual organizations else conformance and interoperability is lost – such as named types.</td></tr>
<tr><td>2</td><td>Rules which may be tailored for individual organizations while still conformant to the NDR structure – such as namespace string contents and namespace tokens.</td></tr>
<tr><td>3</td><td>Rules which may be modified by individual organizations while still conformant to agreed upon data models – such as the use of global or local element declarations.</td></tr>
<tr><td>4</td><td>Rules that if violated loose conformance with the CEFACT data/process model – such as <code>xsd:redefine</code>, <code>xsd:any</code>, and <code>xsd:substitutionGroups</code>.</td></tr>
<tr><td>5</td><td>Rules that relate to extension that are not used by UN/CEFACT and have specific restrictions on their use by other than CEFACT organizations.</td></tr>
<tr><td>6</td><td>Rules that relate to extension that are determined by specific organizations.</td></tr>
<tr><td>7</td><td>Rules that can be modified while not changing instance validation capability.</td></tr>
</table></td><td>1</td></tr>
<tr><td>[R 8059]</td><td>All XML Schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures Second Edition and XML Schema 1.1 Part 2: Datatypes.</td><td>1</td></tr>
</table>

| | | |
|---|---|---|
| [R 935C] | All conformant XML instance documents MUST be based on the W3C suite of technical specifications holding recommendation status. | 1 |
| [R 9224] | XML Schema MUST follow the standard structure defined in Appendix B of this document. | 1 |
| [R A9E2] | Each element or attribute XML name MUST have one and only one fully qualified XPath (FQXP). | 1 |
| [R AA92] | Element, attribute and type names MUST be composed of words in the English language, using the primary English spellings provided in the Oxford English Dictionary. | 1 |
| [R 9956] | LowerCamelCase (LCC) MUST be used for naming attributes. | 1 |
| [R A781] | UpperCamelCase (UCC) MUST be used for naming elements and types. | 1 |
| [R 8D9F] | Element, attribute and type names MUST be in singular form unless the concept itself is plural. | 1 |
| [R BFB0] | Element, attribute and type names MUST be drawn from the following character set: a-z and A-Z. | 1 |
| [R AB19] | XML element, attribute and type names constructed from dictionary entry names MUST NOT include periods, spaces, or other separators; or characters not allowed by W3C XML 1.0 for XML names. | 1 |
| [R 9009] | XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other word truncations, except those included in the defining organizations list of approved acronyms and abbreviations. | 1 |
| [R BFA9] | The acronyms and abbreviations listed by the defining organization MUST always be used in place of the word or phrase they represent. | 1 |
| [R 9100] | Acronyms MUST appear in all upper case except for when the acronym is the first set of characters of an attribute in which case they will be all lower case. | 1 |
| [R B8B6] | Empty elements MUST NOT be used, except when their definition include an identifier attribute that serves to reference another element via schema identity constraints. | 1 |
| [R 984C] | Each organization's XML Schema components MUST be assigned to a namespace for that organization. | 1 |
| [R 8E2D] | The XML Schema namespaces MUST use the following pattern:<br>**URN:** `urn:<organization>:<org hierarchy>[:<org hierarchy level>]*:<schematype>:<context category>:<major>:<status>` | 3 |

| | | |
|---|---|---|
| | **URL:** `http://<organization>/<org hierarchy>[/<org hierarchy level>]*/<schematype>/context category/<major>/<status>`<br><br>Where:<br>• organization – An identifier of the organization providing the standard.<br>• org hierarchy – The first level of the hierarchy within the organization providing the standard.<br>• org hierarchy level – Zero to n level hierarchy of the organization providing the standard.<br>• schematype – A token identifying the type of schema module: data\|codelist\|documentation<br>• context category – The context category [business process] for UN/CEFACT from the UN/CEFACT catalogue of common business processes. Other values may be used by the other organizations.<br>• major – The major version number<br>• status – The status of the schema as: draft\|standard. | |
| [R 8CED] | UN/CEFACT namespaces MUST be defined as Uniform Resource Names. | 3 |
| [R B56B] | Published namespace content MUST NOT be changed unless such change does not break backward compatibility. | 1 |
| [R 92B8] | The XML Schema file name for files other than code lists MUST be of the form <SchemaModuleName>_<Version>.xsd, with periods, spaces, or other separators and the words XML Schema File removed. | 3 |
| [R 8D58] | When representing versioning schemes in file names, the period MUST be represented by a lowercase p. | 3 |
| [R B387] | Every XML Schema file MUST have a namespace declared, using the xsd:targetNamespace attribute. | 1 |
| [R 9354] | A Root XML Schema file MUST be created for each unique business information payload. | 1 |
| [R B3E4] | Each Root XML Schema File MUST be named after the <BusinessInformationPayload> XML Schema File in the documentation within the XML Schema File. | 1 |
| [R 9961] | A Root XML Schema file MUST NOT replicate reusable constructs available in XML Schema files that can be referenced through | 1 |

| | | |
|---|---|---|
| | xsd:include. | |
| [R AA56] | A Business Data Type XML Schema File MUST be created within each context category based namespace. | 1 |
| [R 847C] | The bdt:BusinessDataType XML Schema File MUST be named 'Business Data Type XML Schema File' in the documentation within the XML Schema File. | 1 |
| [R 8238] | One Business Information Entity XML Schema Files MUST be created for the context category that is expressed in the namespace. | 1 |
| [R 8252] | The BusinessInformationEntity XML Schema file MUST be named 'Business Information Entity XML Schema File' by placing the name within the Header documentation section of the file. | 1 |
| [R BD2F] | A Restricted Code Lixt XML Schema File MUST be created for each restricted code list used by a BDT. | 1 |
| [R 942D] | Each Restricted Code List XML Schema File MUST contain enumeration values for both the actual codes and the code values. | 1 |
| [R A62F] | Each Restricted Code List XML Schema File MUST be given a unique name within the namespace it belongs. | 1 |
| [R 8A68] | Cases where code lists are used within the XML Schema, a Code List XML Schema file MUST be created to convey code list enumerations for each code list being used. | 1 |
| [R B443] | Each Common Code List XML Schema File must be given a unique name that represents the name of the code list and is unique within the namespace it belongs. | 1 |
| [R B0AD] | The name of each clm:CodeList XML Schema File as defined in the comment within the XML Schema File MUST be of the form:<br><Code List Agency Identifier\|Code List Agency Name><Code List Identification Identifier\|Code List Name>" - Code List XML Schema File"<br>Where:<br>• Code List Agency Identifier – Identifies the agency that maintains the code list<br>• Code List Agency Name – Agency that maintains the code list<br>• Code List Identification Identifier – Identifies a list of the respective corresponding codes<br>• Code List Name – The name of the code list as assigned by the agency that maintains the code list. | 1 |
| [R B564] | Imported XML Schema Files MUST be fully conformant to category 1, 2, 3, 4 and 7 rules as defined in Rule B998. | 4 |

| | | |
|---|---|---|
| [R 9733] | Imported XML Schema File components MUST be derived using these NDR rules from artifacts that are fully conformant to the latest version of the UN/CEFACT Core Components Technical Specification. | 4 |
| [R 8F8D] | Each xsd:schemaLocation attribute declaration MUST contain a resolvable URL. This may include a relative path reference from the location of the current XML Schema file. | 2 |
| [R BF17] | The xsd:schema version attribute MUST always be declared. | 1 |
| [R 84BE] | The xsd:schema version attribute MUST use the following template:<br><xsd:schema ... version=”Draft” \| ”Standard” _<major>”p”<minor>[”p”<revision>]”><br>Where:<br>• Draft \| Standard – is used based upon the status.<br>• <major> - sequential number of the major version.<br>• <minor> - sequential number of the minor version<br>• <revision> - optional sequential number of the revision. | 2 |
| [R 9049] | Every XML Schema file major version number MUST be a sequentially assigned incremental integer greater then zero. | 1 |
| [R A735] | Minor versioning MUST be limited to declaring new optional XML content, extending existing XML content, or refinements of an optional nature. | 1 |
| [R AFA8] | Minor versions MUST NOT rename existing XML Schema defined artifacts. | 1 |
| [R BBD5] | Changes in minor versions MUST NOT break semantic compatibility with prior versions having the same major version number. | 1 |
| [R 998B] | XML Schema files for a minor version XML Schema MUST incorporate all XML Schema components from the immediately preceding version of the XML Schema file. | 1 |
| [R 8DB4] | The first line in an XML Schema file MUST contain:<br>“<?xml version="1.0" encoding="UTF-8"?>” | 1 |
| [R ABD2] | Every XML Schema File MUST contain a comment that identifies its name immediately following the XML declaration. | 1 |
| [R BD41] | Every XML Schema File MUST contain a comment that identifies its owning agency, version and date immediately following the schema name comment using the format defined in Appendix B-2. | 1 |
| [R A0E5] | The xsd:elementFormDefault attribute MUST be declared and its value set to qualified. | 1 |

| | | |
|---|---|---|
| [R A9C5] | The xsd:attributeFormDefault attribute MUST be declared and its value set to unqualified. | 1 |
| [R 9B18] | The xsd prefix MUST be used in all cases when referring to the namespace http://www.w3.org/2001/XMLSchema as follows: xmlns:xsd=http://www.w3.org/2001/XMLSchema. | 1 |
| [R 90F1] | All required CCTS metadata for ABIEs, BBIEs, ASBIEs, and BDTs must be defined in an XML Schema file. | 1 |
| [R 9623] | The name of the CCTS Metadata XML Schema file will be “Core Components Technical Specification Schema File” and will be defined within the comment within the XML Schema file. | 1 |
| [R 9443] | The CCTS Metadata XML Schema File MUST reside in its own namespace which MUST be defined in accordance with rule 8E2D and assigned the prefix ccts. | 1 |
| [R AD26] | xsd:notation MUST NOT be used. | 1 |
| [R ABFF] | The xsd:any element MUST NOT be used. | 4, 6 |
| [R AEBB] | The xsd:any attribute MUST NOT be used. | 4, 6 |
| [R 9859] | Mixed content MUST NOT be used. | 1 |
| [R 926D] | xsd:substitutionGroup MUST NOT be used. | 4, 6 |
| [R 8A83] | xsd:ID/xsd:IDREF MUST NOT be used. | 1 |
| [R 8E89] | xsd:key/xsd:keyref MUST be used for element referencing. | 1 |
| [R B221] | Supplementary component information MUST be represented as Attributes. | 1 |
| [R AFEE] | User defined attributes MUST only be used for Supplementary components. | 1 |
| [R 8EE7] | Attributes MUST be used rather than elements to serve as identifiers when two elements need to be related to one another via schema identity constraints. | 1 |
| [R 9FEC] | An xsd:attribute that represents a supplementary component with variable information MUST be based on an appropriate XML Schema built-in simpleType. | 1 |
| [R B2E8] | A xsd:attribute that represents a supplementary component which uses codes MUST be based on the xsd:simpleType of the appropriate code list. | 1 |
| [R 84A6] | A xsd:attribute that represents a supplementary component which uses | 1 |

| | | |
|---|---|---|
| | identifiers MUST be based on the xsd:simpleType of the appropriate identifier scheme. | |
| [R BCD6] | Every BBIE leaf element declaration MUST be of the BusinessDataType that represents the source basic business information entity (BBIE) data type. | 1 |
| [R 8337] | The xsd:nillable attribute MUST NOT be used. | 1 |
| [R 8608] | Anonyms types MUST NOT be used. | 1 |
| [R A4CE] | An xsd:complexType MUST be defined for each CCTS BIE. | 1 |
| [R BC3C] | An xsd:complexType MUST be defined for each CCTS BDT that cannot be fully expressed using an xsd:simpleType. | 1 |
| [R A010] | The xsd:all element MUST NOT be used. | 1 |
| [R AB3F] | xsd:extension MUST only be used in the Business DataType XML Schema file. | 4 6 |
| [R 9D6E] | xsd:extension MUST only be used for declaring xsd:attributes to accommodate relevant supplementary components. | 4 6 |
| [R 8AF7] | When xsd:restriction is applied to a xsd:simpleType or xsd:complexType that represents a data type the derived construct MUST use a different name. | 1 |
| [R 847A] | Each defined or declared construct MUST use the xsd:annotation element for required CCTS documentation. | 1 |
| [R 88DE] | Usage rules whose ccts:ConstraintType is something other than “unstructured” MUST be expressed within a ccts:UsageRule element within an xsd:documentation element. | 1 |
| [R B851] | The structure of the ccts:ConstraintType element MUST be:<br>• ccts:UniqueID [1..1]<br>• ccts:Constraint [1..1]<br>• ccts:ConstraintType [1..1]<br>• ccts:ConditionType [1..1<br>• ccts:Name [0..1]<br>• ccts:BusinessTerm [0..*] | 1 |
| [R A1CF] | Usage rules whose ccts:ConstraintType is unstructured MUST be expressed within a ccts:UsageRule element within an xsd:documentation element. | 1 |
| [R B96F] | The Root XML Schema file MUST be assigned to a unique namespace | 1 |

| | | |
|---|---|---|
| | token that represents the context category value it is intended. | |
| [R B698] | The Root XML Schema file MUST include the XML Schema files that are in the same namespace as the Root XML Schema file:<br>• BIE XML Schema file<br>• BDT XML Schema file | 1 |
| [R ACBD] | A Root Schema in one namespace that is dependent upon type definitions or element declarations defined in another namespace MUST NOT import XML Schema Files from that namespace. | 1 |
| [R BD9F] | A global element known as the root element, representing the business information payload, MUST be declared in the Root XML Schema File. | 1 |
| [R A466] | The name of the root element MUST be the name of the business information payload with separators and spaces removed. | 1 |
| [R 8062] | The root element declaration MUST be defined using xsd:complexType that represents the definition of the business information payload. | 1 |
| [R 8837] | Each Root XML Schema File MUST define a xsd:complexType that fully describes the business information payload. | 1 |
| [R 9119] | The name of the root schema xsd:complexType MUST be the name of the root element with the word 'Type' appended. | 1 |
| [R BA43] | For each referenced ABIE element one xsd:unique constraint involving the identifier attribute of the referenced element MUST be declared in the schema, under the scoping element. | 1 |
| [R B40C] | The name of the xsd:unique constraint MUST be composed as follows:<br>"<Scoping Element Name Text><Referenced Element Name Text>Key"<br>So that the name is unique in the schema. This declaration will guarantee uniqueness of the identifier attribute values across all referenced elements of the same name, in the given scope.<br>Where:<br>• Scoping Element Name Text – is the element name within XML document hierarchy which a closed set of reference is defined.<br>• Referenced Element Name Text – is the element name within the scoping element being referenced. | 1 |
| [R AC2D] | For each referenced element in a given scope one xsd:keyref constraint involving the reference attribute that point to the referenced element MUST be declared in the XML Schema, under the scoping element. | 1 |
| [R 9BE8] | Since the XML Schema will specify which parent element can contain the reference attribute, there MUST only be one xsd:keyref constraint | 1 |

| | | |
|---|---|---|
| | declared for all the elements where the reference attribute may occur. | |
| [R 858D] | The name of the xsd:keyref constraint MUST conventionally be composed as follows:<br>"<Scoping Element Name Text><Reference Attribute Name Text>"<br>So that the name is unique in the schema where:<br>• Scoping Element Name Text – is the element name within XML document hierarchy which a closed set of reference is defined.<br>• Reference Attribute Name Text – is the element name within the scoping element being referenced. | 1 |
| [R 886A] | Uniqueness of @key attributes that are not involved in structural referencing MUST NOT be enforced by the schema via identity constraints. Uniqueness of @key attributes should be assured by use of adequate algorithms for the generation of the identifiers (e.g. UUIDs). | 1 |
| [R 8010] | The Root XML Schema File root element declaration MUST have a structured set of annotations documentation present in that includes:<br>• UniqueID (mandatory): The identifier that references the business information payload instance in a unique and unambiguous way.<br>• VersionID (mandatory): The identifier that reference the version of the business information payload instance.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be RSM.<br>• Name (mandatory): The name of the business information payload.<br>• Definition (mandatory): A brief description of the business information payload.<br>• BusinessTermName (mandatory): The business term name that the payload object is known by. | 1 |
| [R 8FE2] | The Business Information Entity XML Schema file MUST include the Business Data Type XML Schema File that resides in the same namespace. | 1 |
| [R AF95] | For every object class (ABIE) identified in the corresponding syntax-neutral model, a named xsd:complexType MUST be defined. | 1 |
| [R 9D83] | The name of the ABIE xsd:complexType MUST be the ccts:DictionaryEntryName: with the spaces and separators removed, approved abbreviations and acronyms applied and with the 'Details' suffix replaced with 'Type'. | 1 |
| [R 9C70] | Every aggregate business information entity (ABIE) xsd:complexType definition content model MUST use zero or more xsd:sequence and/or zero or more xsd:choice elements to reflect each property (BBIE or | 1 |

| | | |
|---|---|---|
| | ASBIE) of its class. | |
| [R 81F0] | Repeating series of only xsd:sequence MUST NOT occur. | 1 |
| [R 8FA2] | Repeating series of only xsd:choice MUST NOT occur. | 1 |
| [R 90F9] | The order and cardinality of the elements within an ABIE xsd:complexType MUST be according to the structure of the ABIE as defined in the model. | 1 |
| [R 8EA2] | Every aggregate business information entity (ABIE) xsd:complexType definition MUST contain an optional “key” attribute that MAY be used as the complex element identifier in a message instance. | 1 |
| [R 92C0] | The “key” attribute MUST be locally define on the ABIE xsd:complexType definition. “key” MUST be a reserved attribute name. | 1 |
| [R 8A37] | Every “key” local attribute MUST be of the type xsd:token. | 1 |
| [R 9DA0] | For each ABIE, a named xsd:element MUST be globally declared. | 1 |
| [R 9A25] | The name of the ABIE xsd:element MUST be the ccts:DictionaryEntryName with the separators and ‘Details’ suffix removed and approved abbreviations and acronyms applied. | 1 |
| [R B27B] | Every ABIE global element declaration MUST be of the xsd:complexType that represents the ABIE. | 1 |
| [R 89A6] | For every attribute of an object class (BBIE) identified in an ABIE, a named xsd:element MUST be locally declared within the xsd:complexType representing that ABIE. | 1 |
| [R AEFE] | Each BBIE element name declaration MUST be the property term and qualifiers and the representation term of the basic business information entity (BBIE). | 1 |
| [R 96D9] | Each BBIE element name declaration where the word ‘identification’ is the final word of the property term and the representation term is ‘identifier’, the term ‘identification’ MUST be removed. | 1 |
| [R 9A40] | Each BBIE element name declaration where the word ‘indication’ is the final word of the property term and the representation term is ‘indicator’, the term ‘indication’ MUST be removed from the property term. | 1 |
| [R A34A] | If the representation term of a BBIE is ‘text’, ‘text’ MUST be removed from the name of the element or type definition. | 1 |
| [R 9025] | For every ASBIE whose ccts:AggregationKind is a composition, a named xsd:element MUST be locally declared. | 1 |
| [R A08A] | For each locally declared ASBIE, the element name MUST be the ASBIE | 1 |

| | | |
|---|---|---|
| | property term and qualifier term(s) and the object class term and qualifier term(s) of the associated ABIE. | |
| [R B27C] | For each locally declared ASBIE, the element declaration MUST use the xsd:complexType that represents its associated ABIE. | 1 |
| [R 9241] | For every ASBIE whose AggregationKind is shared, where the association is implemented as a nested property, the globally declared element for the associated ABIE MUST be referenced using xsd:ref. | 1 |
| [R B78E] | Every ASBIE whose AggregationKind is not a composition, and where the association must be implemented as a referenced property, an equivalent referencing element pointing to the associated ABIE MUST be locally declared. | 1 |
| [R AEDD] | The equivalent referencing element MUST have a name composed of the ASBIE property term and property qualifier term(s). | 1 |
| [R B173] | For each equivalent referencing element a xsd:complexType MUST be declared. Its structure will be an empty element with a local attribute. | 1 |
| [R B523] | The name of the local attribute that is part of the empty element MUST be composed of the object class term and object qualifier term(s) of the ABIE being referenced, followed by the suffix 'Reference'. | 1 |
| [R 8B0E] | The name of the xsd:complexType representing the equivalent referencing element MUST be composed of the object class term and object qualifier term(s) of the ABIE being referenced, followed by the suffix 'ReferenceType'. | 1 |
| [R B7D6] | Each equivalent referencing element MUST be of the xsd:complexType that relates to the ABIE being referenced. | 1 |
| [R ACB9] | For every ABIE xsd:complexType definition a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An identifier of the evolution over time of an ABIE instance.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be ABIE.<br>• DictionaryEntryName (mandatory): The official name of an ABIE.<br>• Definition (mandatory): The semantic meaning of an ABIE.<br>• ObjectClassName (mandatory): The Object Class Name of the ABIE.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object. | 1 |

| | | |
|---|---|---|
| [R 88B6] | For every ABIE xsd:element declaration definition, a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An identifier of the evolution over time of an ABIE instance.<br>• CCTSArtifact (mandatory): The abbreviation code of the type of component. In this case the value will always be ABIE.<br>• DictionaryEntryName (mandatory): The official name of an ABIE.<br>• Definition (mandatory): The semantic meaning of an ABIE.<br>• ObjectClassName (mandatory): The Object Class Name of the ABIE.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object. | 1 |
| [R B8BE] | For every BBIE xsd:element declaration a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references a BBIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of a BBIE instance.<br>• SequencingKeyID (mandatory): Identifier of the sequence of the BBIE in the containing ABIE.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be BBIE.<br>• DictionaryEntryName (mandatory): The official name of the BBIE.<br>• Definition (mandatory): The semantic meaning of the BBIE.<br>• Cardinality (mandatory): Indication whether the BIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.<br>• ObjectClassQualifierName (optional): Qualifies the Object Class Name of the parent ABIE.<br>• ObjectClassName (mandatory): The Object Class Name of the parent ABIE.<br>• PropertyQualifierName (mandatory): Qualifies the Property Term of the BBIE.<br>• PropertyTermName (mandatory): The Property Term Name of the BBIE.<br>• RepresentationTermName (mandatory): Representation term.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the | 1 |

| | | |
|---|---|---|
| | Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the BBIE is commonly known and used in the business.<br>• Example (optional, repetitive): Example of a possible value of a BBIE. | |
| [R 926A] | For every ASBIE xsd:element declaration a structured set of annotations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references an ASBIE instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the ASBIE instance.<br>• SequencingKeyID (mandatory): Identifier of the sequence of the ASBIE in the containing ABIE.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be ASBIE.<br>• DictionaryEntryName (mandatory): The official name of the ASBIE.<br>• Definition (mandatory): The semantic meaning of the ASBIE.<br>• Cardinality (mandatory): Indication whether the ASBIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.<br>• ObjectClassQualifierName (optional): A term that qualifies the Object Class Name of the associating ABIE.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object. | 1 |
| [R 8E0D] | The BusinessDataType XML Schema file MUST include the RestrictedCodeList XML Schema files that are defined in the same namespace. | 1 |
| [R B4C0] | The BusinessDataType XML Schema file MUST import the CommonCodeList XML Schema files that it makes use of in the definition of the BDTs. | 1 |
| [R AE00] | Each CCTS BDT artifact within the UN/CEFACT Data Type Catalogue MUST be defined as an xsd:simpleType or xsd:complexType. | 1 |
| [R 973C] | The name of a business data type MUST be its dictionary entry name with separators and spaces removed. | 1 |
| [R 80FD] | Every restricted Business Data Type XML Schema Component xsd:type definition MUST be derived from its base type using xsd:restriction unless a non-standard variation from the base type is required. | 1 |
| [R A9F6] | Every restricted Business Data Type XML Schema Component xsd:type | 1 |

| | | |
|---|---|---|
| | definition requiring a non-standard variation from its base type MUST be derived from the BDT TextType XML Schema component. | |
| [R AA60] | Every business data type based on a single codelist xsd:simpleType MUST contain one of the following:<br>• xsd:restriction element with the xsd:base attribute set to the code lists defined simple type with appropriate namespace qualification or<br>• xsd:union element with, the xsd:base attribute set to the code list defined simple type and the xsd:member type attribute set to the code list defined simple types with appropriate namespace qualification. | 1 |
| [R AAD1] | Every business data type that has a choice of two or more code lists MUST be defined as one of the following:<br>• A xsd:complexType that contains the xsd:choice element whose content model consists of element references for the alternative code lists to be included with appropriate namespace qualification<br>• A xsd:simpleType that contains the xsd:union element whose xsd:memberType includes the simpleType definitions of the alternative code lists to be included with appropriate namespace qualification. | 1 |
| [R 8B3D] | Global xsd:element declarations MUST NOT occur in the BDT XML Schema File. | 1 |
| [R B340] | Global xsd:attribute declarations MUST NOT occur in the BDT XML Schema File. | 1 |
| [R ACA7] | Local xsd:attribute declarations MUST only represent CCTS Supplementary Components for the Business Data Type for which they are being declared. | 1 |
| [R BFE5] | Every business data type definition MUST contain a structured set of annotation documentation in the following sequence and pattern:<br>• UniqueID (mandatory): The identifier that references a Business Data Type instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the Business Data Type instance.<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be BDT.<br>• DictionaryEntryName (mandatory): The official name of the Business Data Type.<br>• Definition (mandatory): The semantic meaning of the Business Data Type. | 1 |

| | | |
|---|---|---|
| | • DataTypeQualifierName (mandatory): A name that qualifies the Representation Term in order to differentiate it from its underlying Core Data Type and other Business Data Type.<br>• DataTypeName (mandatory): Name of the DataType.<br>• PrimitiveTypeCode (mandatory): The primitive data type of the Business Data Type.<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the BDT is commonly known and used in the business.<br>• Example (optional, repetitive): Example of a possible value of a Business Data Type. | |
| [R 9C95] | For every supplementary component xsd:attribute declaration a structured set of annotation documentations MUST be present in the following pattern:<br>• UniqueID (mandatory): The identifier that references a Supplementary Component of a Core Component Type instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the BDT Supplementary Component instance.<br>• SequencingKeyID (mandatory): Identifier of the sequence of the BDT Supplementary Component.<br>• CCTSArtifact (mandatory): The type of component. In this case the value will always be BDTSC.<br>• DictionaryEntryName (mandatory): The official name of the ASBIE.<br>• Definition (mandatory): The semantic meaning of the ASBIE.<br>• DataTypeQualifierName (mandatory):<br>• DataTypeName (mandatory):<br>• PropertyTermName (mandatory): The Property Term Name of the associated Supplementary Component.<br>• RepresentationTermName (mandatory):<br>• PrimitiveTypeCode (mandatory):<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the BDT is commonly known and used in the business.<br>• Example (optional, repetitive): Example of a possible value of a Supplementary Component. | 1 |

<table>
<tr><td>[R 9E40]</td><td>Each UN/CEFACT maintained code list MUST be defined in its own XML Schema file.</td><td>2</td></tr>
<tr><td>[R 849E]</td><td>The schema module file name for code lists and identifier lists, MUST be of the form:<br><br><Agency Identifier | Agency Name Text>_<List Identification Identifier | List Name Text>_<Version Identifier>.xsd<br><br>All periods, spaces, or other separators are removed except for the “.” before xsd and the “_” between the names.<br><br>Where:<ul><li>Agency Identifier = identifies the agency that manages the list. The default agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.</li><li>Agency Name Text = the name of the agency that maintains the list.</li><li>List Identification Identifier = identifies a list of the respective corresponding codes or ids.</li><li>List Name Text = the name of a list of codes.</li><li>Version Identifier = identifies the version.</li></ul></td><td>2</td></tr>
<tr><td>[R 992A]</td><td>The XML Schema namespaces for code list XML Schema files MUST use the following pattern:<br><br><table><tr><td>URN:</td><td>urn:<organization>:<org hierarchy> *[:<org hierarchy level n>]:codelist:common:<major>:<status>:<name></td></tr><tr><td>URL:</td><td>http://<organization>/<org hierarchy>*[/<org hierarchy level n>]/codelist/common/<major>/<status>/<name></td></tr></table><br>Where:<ul><li>organization – Identifier of the organization providing the standard.</li><li>org hierarchy – The first level of the hierarchy within the organization providing the standard.</li><li>org hierarchy level – Zero to n level hierarchy of the organization providing the standard.</li><li>codelist – A fixed value token for common codelists.</li><li>common – A fixed value token for common codelists.</li><li>major – The Major version number of the codelist.</li></ul></td><td>1</td></tr>
</table>

| | | |
|---|---|---|
| | • status – The status of the schema as: draft\|standard<br>• name – The name of the XML Schema file (using upper camel case) with periods, spaces, or other separators and the words 'schema module' removed.<br>o Code list names are further defined as: <Code List Agency Identifier\|Code List Agency Name Text> ><divider><Code List Identification Identifier\|Code List Name Text><br>▪ Where:<br>• Code List Agency Identifier – is the identifier for the agency that code list is from.<br>• Code List Agency Name Text – is the text of the name that the code list is from.<br>• Divider – the divider character for URN is ':' the divider character for URL is '/'.<br>• Code List Identification Identifer – is the identifier for the given code list.<br>• Code List Name Text – is the text of the name for the code list. | |
| [R 9FD1] | Each UN/CEFACT maintained Common Code list XML Schema File MUST be represented by a unique token constructed as follows:<br>clm[<Business data type name>]<Code List Agency Identifier\|Code List Agency Name Text><Code List Identification Identifier\|Code List Name Text><br>Where any repeated words are eliminated.<br>• Business Data Type Name – is the name of the business data type in the business data type XML Schema file.<br>• Code List Agency Identifier – is the identifier for the agency that code list is from.<br>• Code List Agency Name Text – is the text of the name that the code list is from.<br>• Code List Identification Identifier – is the identifier for the given code list.<br>• Code List Name Text – is the text of the name for the code list. | 2 |
| [R 86C8] | Common Code List XML Schema files MUST NOT import or include any other XML Schema Files. | 1 |
| [R A8EF] | In each Common Code List XML Schema File one, and only one, named xsd:simpleType MUST be defined for the content component. | 1 |
| [R 92DA] | In each Common Code List XML Schema File the name of the xsd:simpleType MUST be the name of code list root element with the | 1 |

| | | |
|---|---|---|
| | word 'ContentType' appended. | |
| [R B40B] | In each Common Code List XML Schema File the xsd:restriction element base attribute value MUST be set to xsd:token. | 1 |
| [R 962C] | Each code in a Common Code List MUST be expressed as an xsd:enumeration, where the xsd:value for the enumeration is the actual code value. | 1 |
| [R 8D1D] | In each Common Code List XML Schema File a single root element MUST be globally declared within the given code list XML Schema file. | 1 |
| [R BE84] | In each Common Code List XML Schema File the code list root element MUST be of a type representing the actual list of code values represented by the type whose name ends in 'ContentType'. | 1 |
| [R BFE5] | Every Common Code List MUST contain a structured set of annotation documentation in the following sequence and pattern:<br>• UniqueID (mandatory): The identifier that references a Business Data Type instance in a unique and unambiguous way.<br>• VersionID (mandatory): An indication of the evolution over time of the Code List.<br>• Name (optional):<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be CLM.<br>• Description (mandatory):<br>• PrimitiveTypeCode (mandatory): The primitive data type of the Code List.<br>• ModificationAllowedIndicator (mandatory):<br>• DefaultIndicator (mandatory):<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the Code List is commonly known and used in the business. | 1 |
| [R A814] | Each code list xsd:enumeration MUST contain a structured set of annotations in the following sequence and pattern:<br>• CCTSArtifact (mandatory): The code of the type of component. In this case the value will always be CLM<br>• Content (optional): The code of value for an enumeration.<br>• Name (optional): The name or text that the represents.<br>• Description (optional): Descriptive information concerning the code<br>• UsageRule (optional, repetitive): Indicates the Usage Rule of the | 1 |

| | | |
|---|---|---|
| | Object.<br>• BusinessTermName (optional, repetitive): A synonym term under which the Code List Value is commonly known and used in the business. | |
| [R 9FD1] | Restricted Code List XML Schema file MUST be used to<br>• Extend existing common code list or<br>• Define a codelist where one does not exist or<br>• Restrict the value of a common codelist for the context category in which it is defined. | 2 |
| [R 86C8] | Restrict Code List XML Schema files MUST NOT import or include any other XML Schema files other than possibly a Common Code List XML Schema file which it is restricting. | 1 |
| [R ACE9] | All XML MUST be instantiated using UTF. UTF-8 should be used if possible, if not UTF-16 should be used. | 1 |
| [R A1B9] | The xsi namespace prefix MUST be used to reference the "http://www.w3.org/2001/XMLSchema-instance" namespace and anything defined by the W3C XMLSchema-instance namespace. | 1 |
| [R 9277] | The xsi:nil attribute MUST NOT appear in any conforming instance. | 1 |
| [R 8250] | The xsi:type attribute MUST NOT be used within an XML Instance. | 1 |

# Appendix I. Glossary

**Aggregate Business Information Entity** (ABIE) – A collection of related pieces of business information that together convey a distinct business meaning in a specific business context. Expressed in modelling terms, it is the representation of an object class, in a specific business context.

**Aggregate Core Component (ACC)** – A collection of related pieces of business information that together convey a distinct business meaning, independent of any specific business context. Expressed in modelling terms, it is the representation of an object class, independent of any specific business context.

**Aggregation** – An Aggregation is a special form of Association that specifies a whole-part relationship between the aggregate (whole) and a component part.

**Artefact –** A piece of information that is produced, modified, or used by a process. An artefact can be a model, a model element, or a document. A document can include other documents. CCTS artefacts include all registry classes as specified in Section 9 of the *CCTS Technical Specification* and all subordinate named constructs of a CCTS registry class.

**Assembly Rules –** Assembly Rules group sets of unrefined business information entities into larger artefacts suitable for expressing complete business information exchange concepts.

**Association Business Information Entity (ASBIE)** – A business information entity that represents a complex business characteristic of a specific object class in a specific business context. It has a unique business semantic definition. An Association Business Information Entity represents an Association Business Information Entity property and is therefore associated to an Aggregate Business Information Entity, which describes its structure. An Association Business Information Entity is derived from an Association Core Component.

**Association Business Information Entity Property** – A business information entity property for which the permissible values are expressed as a complex structure, represented by an Aggregate Business Information Entity.

**Association Core Component (ASCC)** – 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. An Association Core Component represents an Association Core Component Property and is associated to an Aggregate Core Component, which describes its structure.

**Association Core Component Property** – A core component property for which the permissible values are expressed as a complex structure, represented by an Aggregate Core Component.

**Attribute** – A named value or relationship that exists for some or all instances of some entity and is directly associated with that instance.

**Backward Compatibility –** Any XML instance that is valid against one schema version will also validate against the previous schema version.

**Basic Business Information Entity (BBIE)** – A business information entity that represents a singular business characteristic of a specific object class in a specific business context. It has a unique business semantic definition. A Basic Business Information Entity represents a Basic Business Information Entity property and is therefore linked to a data type, which describes it values. A Basic Business Information Entity is derived from a Basic Core Component.

**Basic Business Information Entity Property** – A business information entity property for which the permissible values are expressed by simple values, represented by a data type.

**Basic Core Component (BCC)** – A core component which constitutes a singular business characteristic of a specific Aggregate Core component that represents a object class. It has a unique business semantic definition. a Basic Core Component represents a Basic Core Component property and is therefore of a data type, which defines its set of values. Basic core components function as the properties of Aggregate Core components.

**Basic Core Component (BCC) Property** – A core component property for which the permissible values are expressed by simple values, represented by a data type.

**Business Context** – The formal description of a specific business circumstance as identified by the values of a set of context categories, allowing different business circumstances to be uniquely distinguished.

**Business Data Type –** A business data type is a data type, which consists of one and only one BDT content component, that carries the actual content plus one or more BDT supplementary component giving an essential extra definition to the CDT content component. BDTs do not have business semantics.

**Business Data Type Content Component –** Defines the primitive type used to express the content of a core data type.

**Business Data Type Content Component Restriction** – The formal definition of a format restriction that applies to the possible values of a core data type content component.

**Business Data Type Supplementary Component** – Gives additional meaning to the business data type content component.

**Business Data Type Supplementary Component Restrictions** – The formal definition of a format restriction that applies to the possible values of a business data type Supplementary Component.

**Business Information Entity (BIE) –** A piece of business data or a group of pieces of business data with a unique business semantic definition. A business information entity can be a Basic Business Information Entity (BBIE), an Association Business Information Entity (ASBIE), or an Aggregate Business Information Entity (ABIE).

**Business Information Entity (BIE) Property** – A business characteristic belonging to the Object Class in its specific business context that is represented by an Aggregate Business Information Entity.

**Business Libraries** – A collection of approved process models specific to a line of business (e.g., shipping, insurance).

**Business Process** – The business process as described using the UN/CEFACT Catalogue of Common business processes.

**Business Process Context** – The business process name(s) as described using the *UN/CEFACT Catalogue of Common Business Processes* as extended by the user.

**Business Process Role Context** – The actors conducting a particular business process, as identified in the *UN/CEFACT Catalogue of Common Business Processes*.

**Business Semantic(s)** – A precise meaning of words from a business perspective.

**Business Term** – This is a synonym of the dictionary entry name under which the artefact is commonly known and used in business. A CCTS artefact may have several business terms or synonyms.

**Cardinality** – An indication of the minimum and maximum occurences for a characteristic: not applicable (0..0), optional (0..1), optional repetitive (0..*) mandatory (1..1), mandatory repetitive (1..*), fixed (n..n) where n is a non-zero positive integer.

**Catalogue of Business Information Entities** – This represents the approved set of Business Information Entities from which to choose when applying the Core Component discovery process

**CCL** – see Core Component Library.

**Classification Scheme** – This is an officially supported scheme to describe a given context category.

**Composition** – A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. Composition may be recursive.

**Context** – Defines the circumstances in which a business process may be used. This is specified by a set of context categories known as business context.

**Context Category** – A group of one or more related values used to express a characteristic of a business circumstance.

**Controlled Vocabulary** – A supplemental vocabulary used to uniquely define potentially ambiguous words or business terms. This ensures that every word within any of the core component names and definitions is used consistently, unambiguously and accurately.

**Core Component (CC)** – A building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept.

**Core Component Library** – The Core Component Library is the part of the registry/repository in which Core Components shall be stored as registry classes. The Core Component Library will contain all the registry classes.

**Core Component Property** – A business characteristic belonging to the object class represented by an Basic Core Component property or an Association Core Component property.

**Definition** – This is the unique semantic meaning of a core component, business information entity, business context or data type.

**Dictionary Entry Name** – This is the official name of a CCTS-conformant artefact.

**Facet** – A facet is a constraining value that represents a component restriction of a Business Data Type content or supplementary component so as to define its allowed value space.

**Geopolitical Context** – Geographic factors that influence business semantics (e.g., the structure of an address).

**Industry Classification Context** – Semantic influences related to the industry or industries of the trading partners (e.g., product identification schemes used in different industries).

**Information Entity –** A reusable semantic building block for the exchange of business-related information.

**LowerCamelCase (LCC) –** LowerCamelCase is a lexical representation of compound words or phrases in which the words are joined without spaces and all but the first word are capitalized within the resulting compound.

**Message Assembly** – The process whereby Business Information Entities are assembled into a usable message for exchanging business information.

**Naming Convention** – The set of rules that together comprise how the dictionary entry name for CCTS artefacts are constructed.

**Object Class** – The logical data grouping (in a logical data model) to which a data element belongs (ISO11179). The object class is the part of a core component or business information entity dictionary entry name that represents an activity or object.

**Object Class Term** – A component of the name of a core component or business information entity which represents the object class to which it belongs.

**Official Constraints Context** – Legal and governmental influences on semantics (e.g. hazardous materials information required by law when shipping goods).

**Primitive Type** – A primitive type, also known as a base type or built-in type, is the basic building block for the representation of a value as expressed by more complex data types.

**Product Classification Context** – Factors influencing semantics that are the result of the goods or services being exchanged, handled, or paid for, etc. (e.g. the buying of consulting services as opposed to materials).

**Property Term** – A semantically meaningful name for the characteristic of the Object Class that is represented by the core component property. It shall serve as basis for the DEN of the basic and Association Core Components that represents this core component property.

**Qualified Business Data Type** – A qualified business data type contains restrictions on a business data type content or business data type supplementary component(s).

**Qualifier Term** – A word or group of words that help define and differentiate an item (e.g. a business information entity or a business data type) from its associated items

(e.g. from a core component, a core data type, another business information entity or another business data type).

**Registry –** An information system that manages and references artifacts that are stored in a repository. The term registry implies a combination of registry/repository.

**Registry Class** – The formal definition of all the common information necessary to be recorded in the registry by a registry artefact – core component, a business information entity, a data type or a business context.

**Repository –** an information system that stores artifacts.

**Representation Term** – The type of valid values for a Basic Core Component or Basic Business Information Entity.

**Restriction –** restriction is the process of deriving a new data structure from an existing data structure under the following rules:

- you can reduce the cardinality range of any field from the existing data structure;
- you can restrict the range of allowed values for any field with a simple data type (e.g. string, number);
- you can add a semantic restriction which narrows the business scope of any field.

All valid instances of a new restricted data structure must also be valid instances of the existing data structure from which the new data structure was derived.

**Supporting Role Context** – Semantic influences related to non-partner roles (e.g., data required by a third-party shipper in an order response going from seller to buyer.).

**Syntax Binding –** The process of expressing a Business Information Entity in a specific syntax.

**System Capabilities Context** – This context category exists to capture the limitations of systems (e.g. an existing back office can only support an address in a certain form).

**UMM Information Entity –** A UMM information entity realizes structured business information that is exchanged by partner roles performing activities in a business transaction. Information entities include or reference other information entities through associations."

**Unique Identifier** – The identifier that references a registry class instance in a universally unique and unambiguous way.

**UpperCamelCase (UCC) –** UpperCamelCase is a lexical representation of compound words or phrases in which the words are joined without spaces and are capitalized within the resulting compound.

**Usage Rules** – Usage rules describe a constraint that describes specific conditions that are applicable to a component in the model.

**User Community** – A user community is a group of practitioners, with a publicized contact address, who may define Context profiles relevant to their area of business. Users within the community do not create, define or manage their individual context

needs but conform to the community's standard. Such a community should liaise closely with other communities and with general standards-making bodies to avoid overlapping work. A community may be as small as two consenting organizations.

**Version** – An indication of the evolution over time of an instance of a core component, data type, business context, or business information entity.

**XML Schema –** A generic term used to identify the family of grammar based XML document structure validation languages to include the more formal W3C XML Schema Definition Language, ISO 8601 Document Type Definition, or Schematron. An XML Schema is a collection of schema components.

**XML Schema Definition Language Component –**The 13 building blocks that comprise the abstract data model of the schema, consisting of simple type definitions, complex type definitions, attribute declarations, element declarations, attribute group definitions, identity-constraint definitions, model group definitions, notation declarations, annotations, model groups, particles, wildcards, and attribute uses.

**XML Schema Definition Language –** The World Wide Web Consortiums official recommendation for describing the structure and constraining the contents of XML documents.

**XML Schema Document –** An XML conformant document expression of an XML schema.

# Disclaimer

The views and specification expressed in this document are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this design.

# Copyright Statement