United Nations Centre for Trade Facilitation and Electronic Business

UN/CEFACT XML Naming and Design Rules

Version 3.0

Draft for the 2nd Public Review of ODP5

30 December 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 XML 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 ODP 5 Draft 2nd Public Review Internal Review 3 of 18 November 2008.

Previous version: UN/CEFACT XML Naming and Design Rules, Version 3.0 ODP 5 Draft 2nd Public Review Internal Review 2 of 4 November 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 *thisUnited Nations Centre For Trade Facilitation and Electronic Business (UN/CEFACT) XML Naming and Design Rules* 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 AG
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, SAP 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 of this Document	Informative
Section 2: Project Team	Informative
Section 3: Introduction	Informative
Section 4: Objectives	Normative
Section 5: XML Schema Architecture	Normative
Section 6: Application of Context	Informative
Section 7: General XML Schema Language Conventions	Normative
Section 8: XML Schema Files	Normative
Section 9: XML Instance Documents	Normative
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: Common Use Cases for Code Lists	Informative
Appendix I: Alternate Message Assembly	Informative
Appendix J: Naming and Design Rules List	Normative
Appendix K: 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.

[R n] – 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 is:

- 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 Core Component Technical Specification (CCTS) conformant artifacts 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 modeling 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.

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

[R B998]	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:	1

Rule Categorization	
ID	Description
1	Rules which must not be violated by individual organizations else conformance and interoperability is lost – such as named types.
2	Rules which may be modified by individual organizations while still conformant to the NDR structure – such as namespace string contents and namespace tokens.
3	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. (Changes to the XML Schema Architecture.)
4	Rules that if violated lose conformance with the UN/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 UN/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.

4.3 Caveats and Assumptions

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

Although this specification defines schema components as expressions of core component artifacts, 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 defintions 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.
- Interchange and Application Use – UN/CEFACT XML Schema and the resulting XML instance documents are intended for a variety of data exchanges.
- 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 is 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 and the corresponding text related to general XML Schema construction including:

- Overall XML Schema Structure
- Relationship to CCTS
- Business Message Syntx Binding
- Naming and Modeling Constraints
- Reusability Scheme
- Namespace Scheme
- XML Schema Files
- Schema Location
- Versioning Scheme

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 and tool support. Accordingly, all UN/CEFACT XML Schema definitions will be expressed in XML Schema. All references to W3C XML Schema will be as XML Schema. References to XML Schema defined by UN/CEFACT 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 Part 2: Datatypes Second Edition.	1

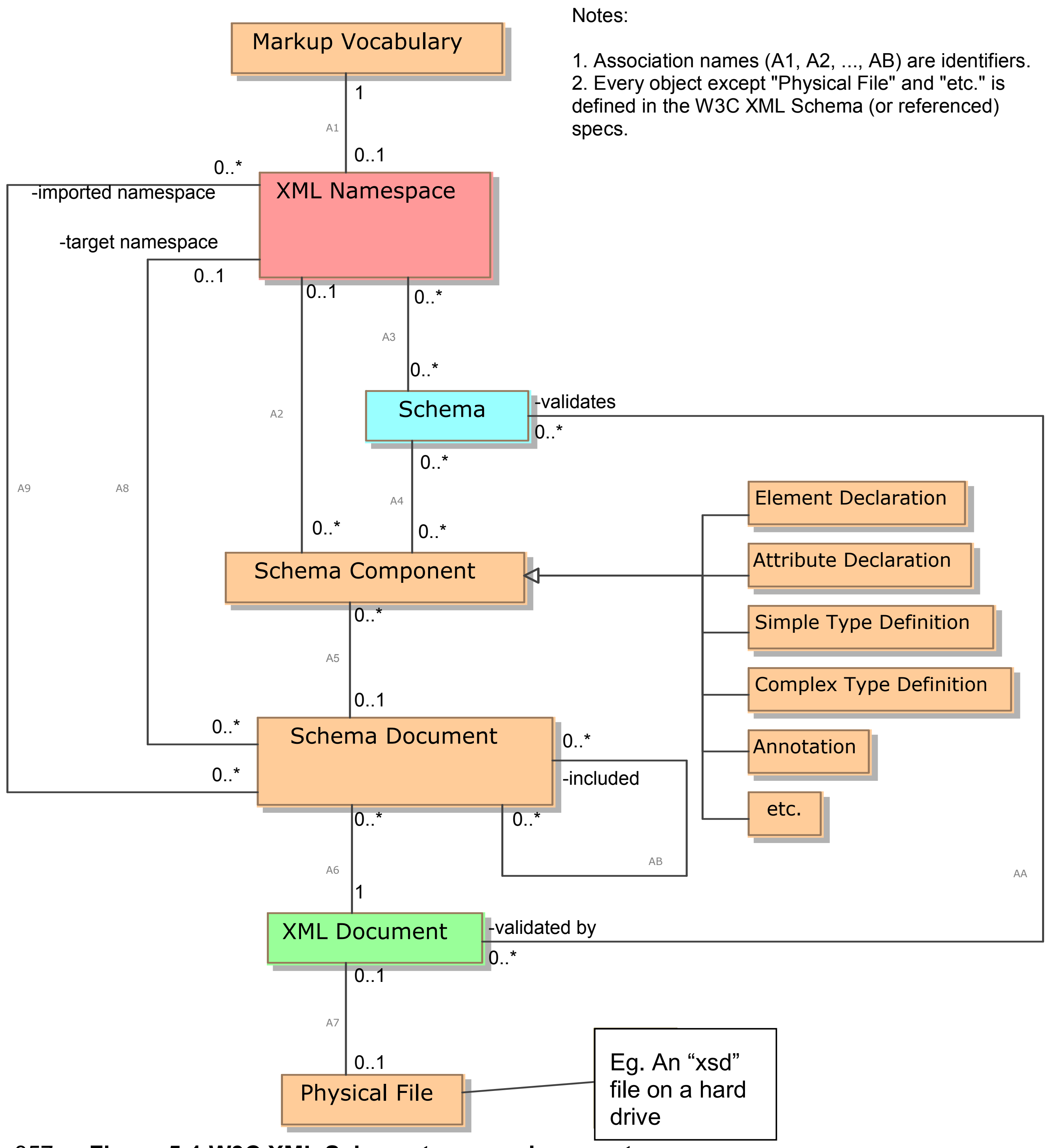
The W3C is the recognized source for XML specifications. W3C specifications can hold various statuses. Only those W3C specifications holding recommendation status are considered by the W3C to be stable specifications.

[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 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 to define 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 terms and concepts and their relationship as defined by the W3C.



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

358 5.2 Relationship to CCTS

359 All UN/CEFACT business information modeling and business process modeling
360 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 interoperability.

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 modeling, information modelling and information exchange. Core components are conceptual models that are used to define Business Information Entities (BIEs).

BIEs are logical data model artefact expressions. BIEs are used for creating interoperable business process models, business documents, and information exchanges. BIEs are created through the application of context to a CC that may:

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

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

The CCTS model for BIEs includes:

- Common Information – information that is expressed in the annotation documentation in the XML Schema.
- Localized Information – information that while expressed in the model is not expressed in the XML Schema.
- Usage Rules – information that is 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. Thus, 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 relevant CCTS CC artifacts, BIE artifacts and XML Schema Components.

[Note:]

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

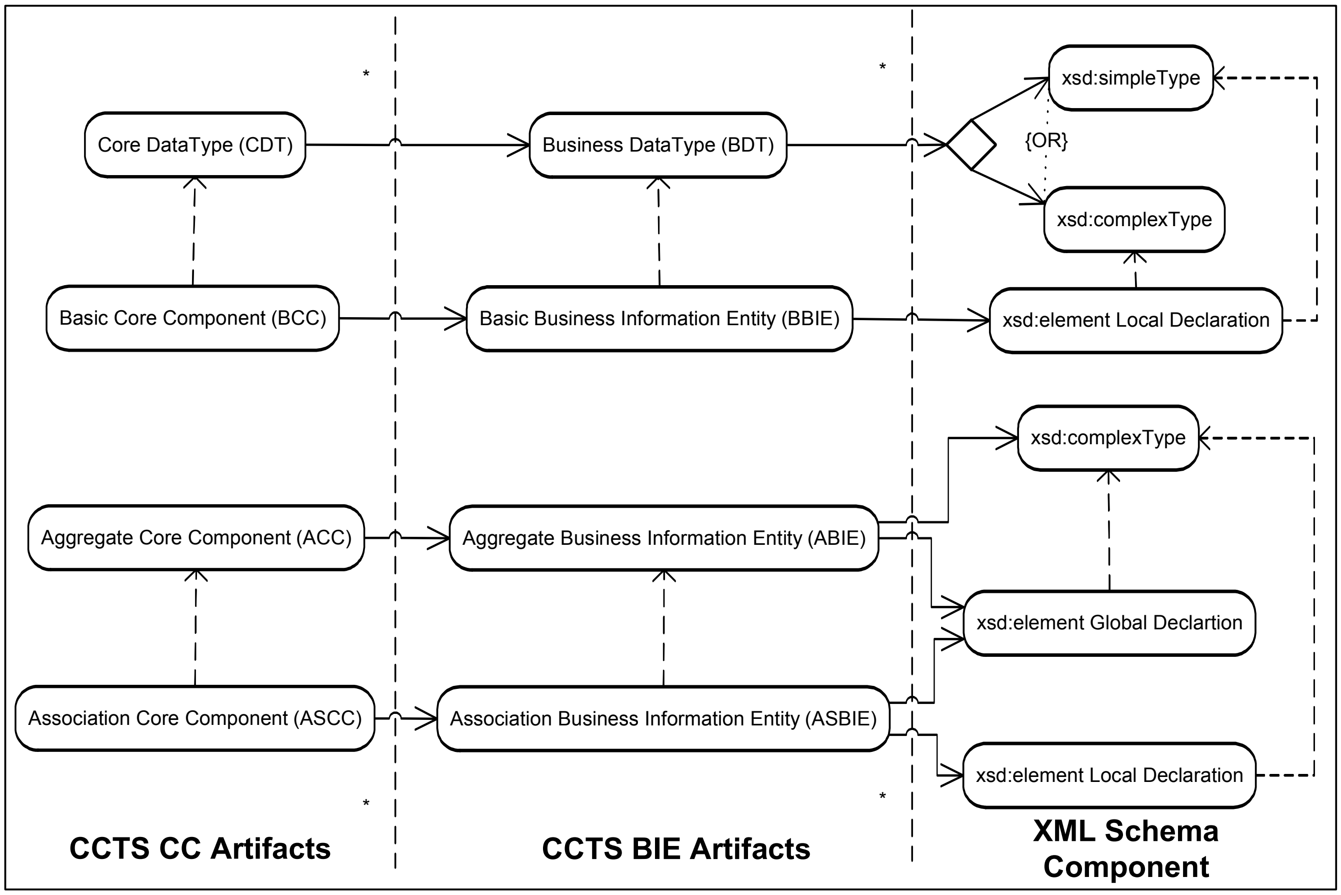

Core DataType (CDT)
Business DataType (BDT)
xsd:simpleType
{OR}
xsd:complexType
Basic Core Component (BCC)
Basic Business Information Entity (BBIE)
xsd:element Local Declaration
xsd:complexType
Aggregate Core Component (ACC)
Aggregate Business Information Entity (ABIE)
xsd:element Global Declartion
Association Core Component (ASCC)
Association Business Information Entity (ASBIE)
xsd:element Local Declaration
CCTS CC Artifacts
CCTS BIE Artifacts
XML Schema Component

Figure 5-2 Transitions between CCTS Artifacts and XML Schema Components

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

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 the artefact type pointed to by the arrow.

5.2.2.1 Aggregate Business Information Entity

All Aggregate Business Information Entities (ABIEs) are represented as a type definition (**`xsd:complexType`**) and global element (**`xsd:element`**) declaration in the UN/CEFACT BIE XML Schema File for the namespace in which they are defined. See section 8.3 Business Information Entities XML Schema Files.

5.2.2.2 Association Business Information Entity

Whether an Association Business Information Entity (ASBIE) uses a local or global element depends upon the type of association (**`AggregationKind=shared`** or **`AggregationKind=composite`**) 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=composite`**). 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=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.

See section 8.3 Business Information Entities XML Schema Files.

5.2.2.3 Basic Business Information Entity

A Basic Business Information Entity (BBIE) is declared as a local element within the **xsd:complexType** representing the parent ABIE. The BBIE is based on a (is of type) BDT. See section 8.3 Business Information Entities XML Schema Files.

5.2.2.4 Business Data Type

A Business Data Type (BDT) is defined as either an **xsd:complexType** or **xsd:simpleType**. If the BDT value domain can be expressed by the facets of an xsd built in data type, then the BDT will be defined as an **xsd:simpleType** whose **xsd:base** is the xsd built in type.

If not, then an **xsd:complexType** will be defined with a content model to support the value domain.

See section 8.4 Business Data Type XML Schema Files.

5.2.3 Context Categories

The CCTS identifies a set of context categories – such as business process, geopolitical, system capabilities, business process role – the values of these categories collectively define the context in which context specific BIEs are defined. This NDR specification captures the context through the use of an annotation application information element (**<xsd:annotation> <xsd:appInfo>**) accompanying each element declaration. See section 7.5.2 Application Information (AppInfo) for more information.

UN/CEFACT uses the business process context value to create different namespaces. 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. See section 6 Application of Context.

5.3 Business Message Syntax Binding

UN/CEFACT will create the XML syntax binding of its CCTS conformant BIE data models directly from the associations and hierarchies expressed in the Business Message Template for each business message exchange. This transformation approach is based on traditional nesting of all components of the data model.

Figure 5-3 shows the UN/CEFACT Business Message structure as defined in the Business Message Template. The Business Message structure consists of a single Message Assembly (MA) component representing the Business Message. Each Association Message Assembly (ASMA) is a proxy for the first level ABIE in a given Business Message. Additionally, application specific information unique to the instance can be defined using the UN/CEFACT Standard Business Document Header specification.

The XML Schema Specification also supports an alternative to nesting. This alternative – using schema identity constraints (`xsd:key`/`xsd:keyRef` – enables referencing and reuse of a given XML element in instance documents. UN/CEFACT is currently evaluating this alternative for future use to include a method for application at the data model level. In anticipation that the data model issues will be resolved, UN/CEFACT has already developed a set of rules for its XML implementation. These rules and the supporting narrative can be found in Appendix I Alternative Business Message Syntax Binding. Organizations using this alternative method will still be considered conformant to this specification, if they adhere to all other conformance requirements and use the rules defined in the Appendix I Alternative Business Message Syntax Binding.

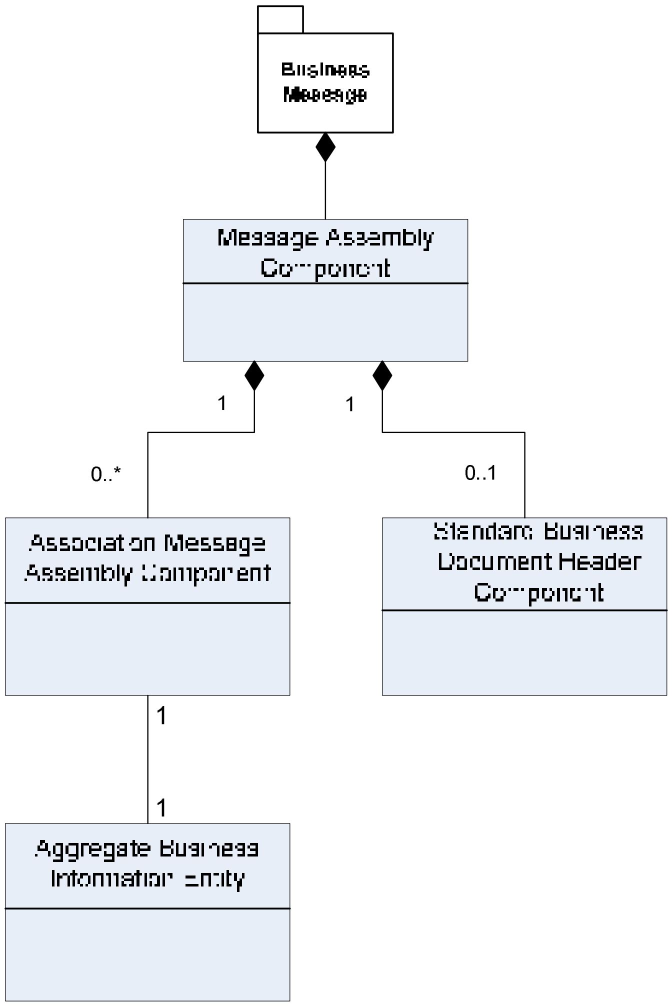

Business Message
Message Assembly Component
1
1
0..*
0..1
Association Message Assembly Component
Standard Business Document Header Component
1
1
Aggregate Business Information Entity

Figure 5-3 Business Message Template Metamodel

The business message MA component is defined as a global type and declared as the sole global element in the Root XML Schema File. The MA content model consists of a set of ASMA element declarations whose type is the xsd:complexType definition in the BIE XML Schema File that represent the first level ABIEs used in the message. It may also contain an optional Standard Business Document Header component. See section 8.2 Root XML Schema Files.

5.4 Naming and Modeling Constraints

UN/CEFACT XML Schemas are derived from components created through the application of CCTS.UN/CEFACT XML Schema contain XML Schema Components that follow the naming and design rules in this specification.

These naming and design rules take advantage of the features of the XML Schema specification. In many cases this approach 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.

The CCTS DEN can be reconstructed by using XPath expressions. The fully qualified XPath (FQXP) ties the information to its standardized CCTS semantics, while the XML element or attribute name is a truncation that reflects the hierarchy of the XML construct.

The FQXP anchors the use of a construct to a particular location in a business information payload. The DEN 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 provides 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

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 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 Instances 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

LowerCamelCase (LCC) is used for naming XML Schema attributes and UpperCamelCase (UCC) is used for naming XML Schema 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.

[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

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" ...>
```

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

[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

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 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

5.5 Reusability Scheme

UN/CEFACT is committed to an object based approach for its process, data, and information models.

UN/CEFACT considered adopting an XSD type based approach which uses 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 5-4) are declared as a **`xsd:complexType`**.
- All attributes of a class are declared as a local **`xsd:element`** within the corresponding **`xsd:complexType`**.
- UML **`aggregationKind=composite`** associations will result in a locally declared **`xsd:element`** with a globally declared **`xsd:complexType`** (e.g. Invoice_Trade.Line.Item and Billed_Delivery in Figure 5-4). A composite aggregation ASBIE represents a relationship wherein if the associationg ABIE ceases to exist the associated ABIE ceases to exist.
- UML **`aggregationKind=shared`** associations will result in a globally declared **`xsd:element`** with a globally declared **`xsd:complexType`** (e.g. Invoice.Buyer. Buyer_Party, Invoice. Seller. SellerParty in Figure 5-4). A shared aggregation ASBIE represents a relationship wherein if the associating ABIE ceases to exist, the associated ABIE continues to exist.

The rules pertaining to the ‘hybrid approach’ are contained in sections 8.3.3 Type Definitions and 8.3.4 Element Declarations and References.

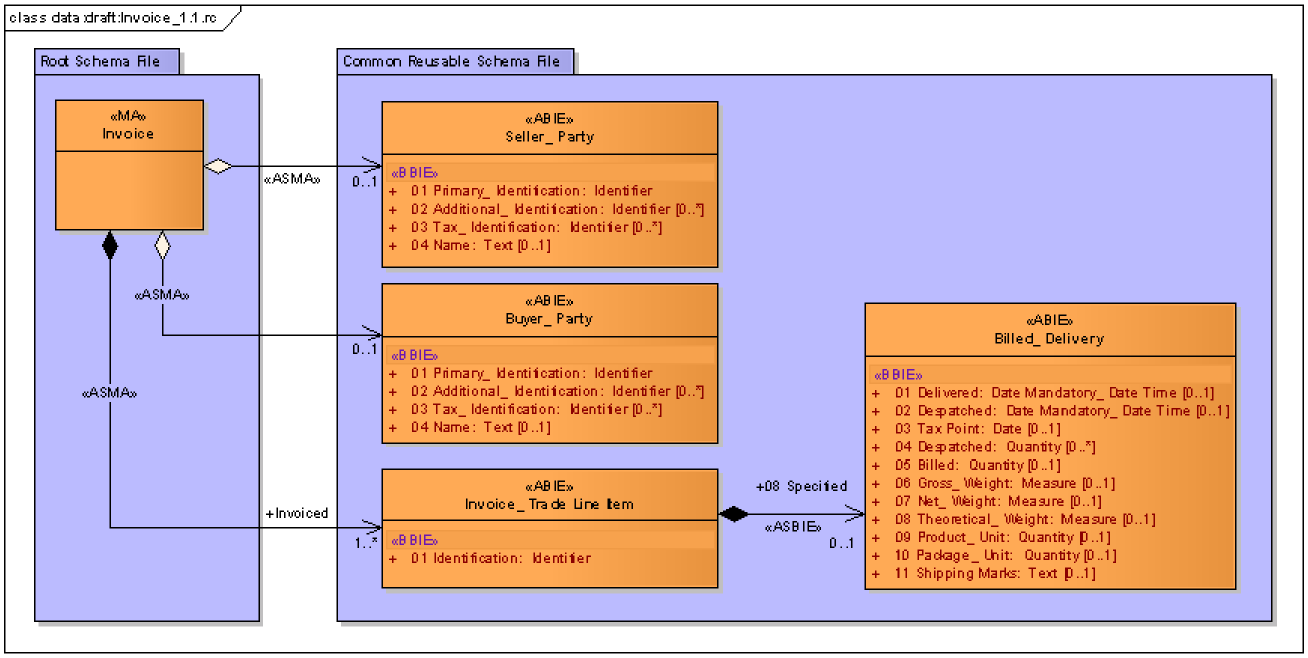

class data:draft:Invoice_1.1.rc
Root Schema File
Common Reusable Schema File
«MA» Invoice
«ASMA»
«ASMA»
«ASMA»
0..1
0..1
+Invoiced
1..*
«ABIE» Seller_Party
«BBIE»
+ 01 Primary_Identification: Identifier
+ 02 Additional_Identification: Identifier [0..*]
+ 03 Tax_Identification: Identifier [0..*]
+ 04 Name: Text [0..1]
«ABIE» Buyer_Party
«BBIE»
+ 01 Primary_Identification: Identifier
+ 02 Additional_Identification: Identifier [0..*]
+ 03 Tax_Identification: Identifier [0..*]
+ 04 Name: Text [0..1]
«ABIE» Invoice_Trade Line Item
«BBIE»
+ 01 Identification: Identifier
+08 Specified
«ASBIE»
0..1
«ABIE» Billed_Delivery
«BBIE»
+ 01 Delivered: Date Mandatory_Date Time [0..1]
+ 02 Despatched: Date Mandatory_Date Time [0..1]
+ 03 Tax Point: Date [0..1]
+ 04 Despatched: Quantity [0..*]
+ 05 Billed: Quantity [0..1]
+ 06 Gross_Weight: Measure [0..1]
+ 07 Net_Weight: Measure [0..1]
+ 08 Theoretical_Weight: Measure [0..1]
+ 09 Product_Unit: Quantity [0..1]
+ 10 Package_Unit: Quantity [0..1]
+ 11 Shipping Marks: Text [0..1]

Figure 5-4 UML Model Example

Figure 5-4 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 following the rules defined in this specification.

[Note] - Tokens

The tokens rsm, bie, bdt, bcl, and ccl are used throughout this document to generically represent root XML Schema Files, BIE XML Schema Files, BDT XML Schema Files, Business Code List XML Schema Files, and Common Code List XML Schema Files. The actual tokens are developed using the rules stated elsewhere in this specification.

Example 5-9: XML Schema declarations representing Figure 5-4.

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

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

 <xsd:complexType name="InvoiceType">
         <xsd:sequence>
                <xsd:element name="ID" type="bdt:IDType"/>
                <xsd:element ref="bie:SellerParty"/>
                <xsd:element ref="bie:BuyerParty"/>
                <xsd:element name="InvoiceTradeLineItem"
type="bie: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="bie: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>
```

5.6 Namespace Scheme

A namespace is an abstract container for a collection of elements, attributes and types that serve to uniquely identify this collection from other collections.

“An 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 UNCEFACT namespaces following the namespace scheme 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 the primary context category value in which the XML Schema is defined similar to the UN/CEFACT namespace scheme shown in Figure 5-5.

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

[Note:]

The primary context category expressed in the namespace may be chosen by the organization defining or publishing the given set of XML Schema Files.

UN/CEFACT has choosen to use the Business Process context category and. UN/CEFACT XML Schema Files will be expressed within a namespace that reflects the Business Process Value that the CCTS artifacts in which the contained XML Schema Components are derived are defined.

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

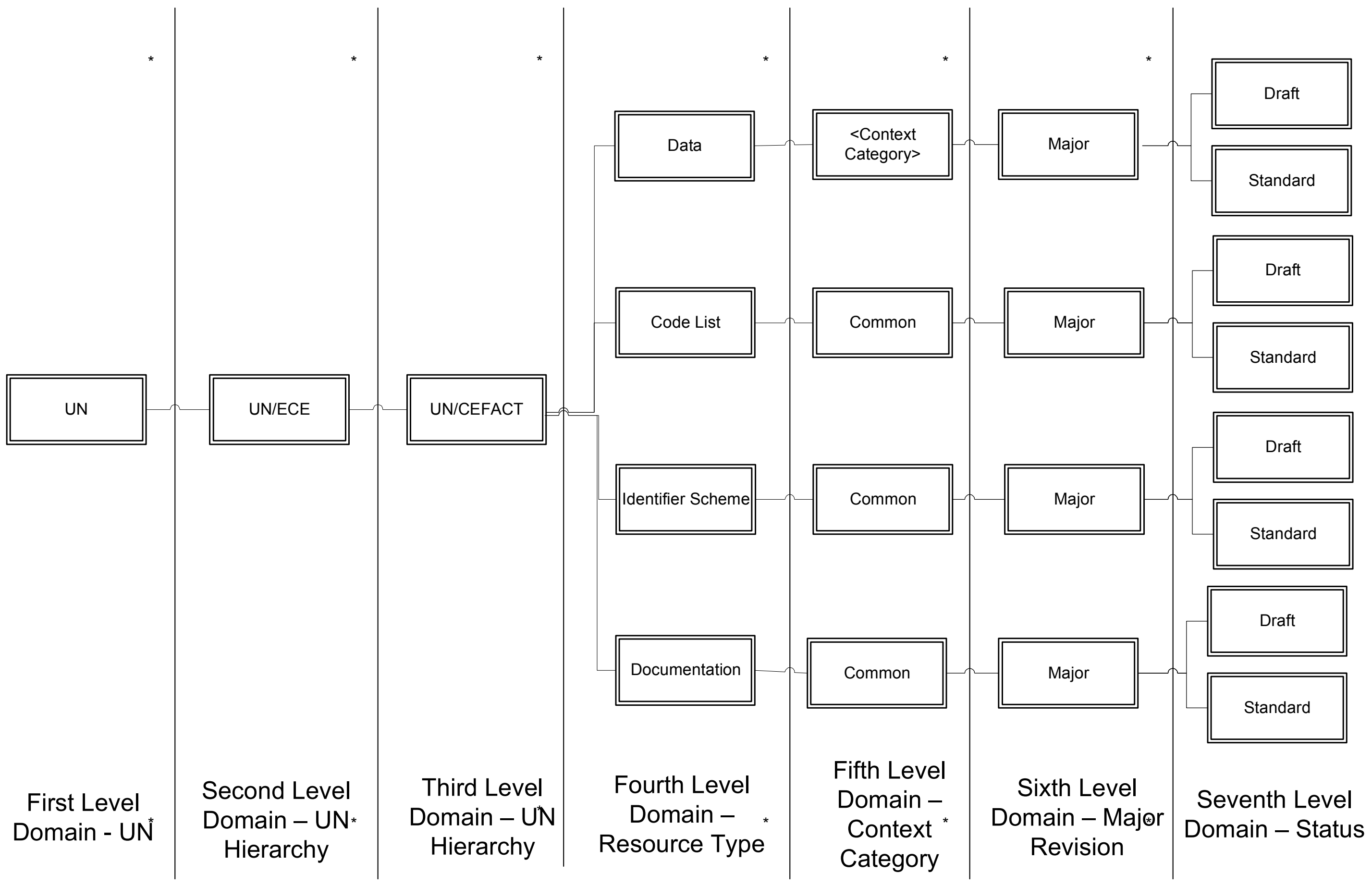

UN
UN/ECE
UN/CEFACT
Data
Code List
Identifier Scheme
Documentation
<Context Category>
Common
Common
Common
Major
Major
Major
Major
Draft
Standard
Draft
Standard
Draft
Standard
Draft
Standard
First Level Domain - UN
Second Level Domain – UN Hierarchy
Third Level Domain – UN Hierarchy
Fourth Level Domain – Resource Type
Fifth Level Domain – Context Category
Sixth Level Domain – Major Revision
Seventh Level Domain – Status

Figure 5-5: UN/CEFACT Namespace Scheme

5.6.1 Namespace Uniform Resource Identifiers

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

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

The URN format will be:

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

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

The URL format will be:

```
http://<organization>/<org hierarchy>[/<org hierarchy
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`**.

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 specification.

[R 8E2D]	The XML Schema namespaces MUST use the following pattern: **URN:** **`urn:<organization>:<org hierarchy>[:<org hierarchy level>]*:<schematype>:<context category>:<major>:<status>`** **URL:** **`http://<organization>/<org hierarchy>[/<org hierarchy 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`**.	3

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 specification. However, each organization must decide for themselves if persistence or resolvability is more important for their namespace solution.

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

Example 5-10 and 5-11 show namespace using URNs that follow the valid format for Draft and Standard specifications.

Example 5-10: Namespace Name at Draft Status

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

Example 5-11: Namespace Name at Specification Status

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

UN/CEFACT namespace names include a major version identifier, therefore once a namespace’s content is published; any change that breaks backward compatibility requires a new namespace. See the section on 5.9.1 Major Versions. Only the publisher of a namespace may change the content defined within the namespace. The publisher may only make changes that adhere to the rules defined for minor version changes defined in section 5.9.2 Minor Versions.

[R B56B]	Published namespace content MUST only be changed by the publishing organization of the namespace or its successor.	1

5.6.2 Namespace Tokens

Namespace URIs are typically aliased using tokens rather than citing the entire URI for the qualifier in a qualified name for XML Schema Components within a given namespace.

Namespace tokens representing the namespace will be created using three character representations for each unique value within the chosen context category.

Additionally, XML Schema Files that are defined for Common Code List will use a token that is prefixed with ‘clm’ to indicate that they are Common Code List XML Schema Files.

5.7 XML Schema Files

An 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 Components that collectively comprise an abstract data model.

For consistency, XML Schema File names will adhere to a specific pattern.

[R 92B8]	The XML Schema File name for files other than code lists and identifier schemes 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

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 thoughtout approach to the layout provides an efficient and effective mechanism for providing components as needed rather than dealing with complex, multi-focused XML Schema Files. XML Schema Files created from this specification represent abstract data models for messages, CCTS conformant ABIEs, BDTs, Business Code Lists (BCL), Business Identifier Schemes (BIS), references to Common Code Lists (CCL) and Common Identifier Schemes (CIS). Figure 5-6 shows how these schema files are collected into relevant namespaces representing business processes/information messages.

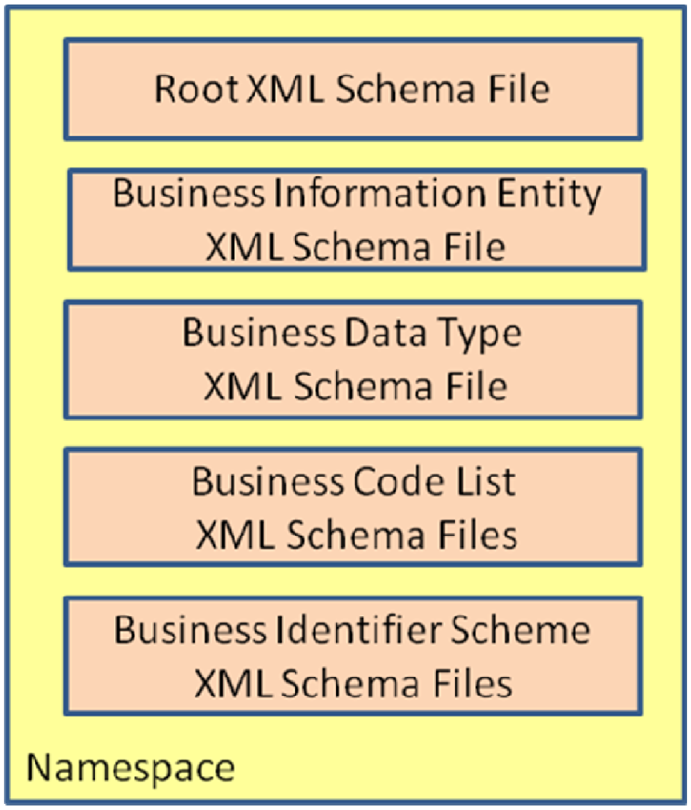

Root XML Schema File
Business Information Entity XML Schema File
Business Data Type XML Schema File
Business Code List XML Schema Files
Business Identifier Scheme XML Schema Files
Namespace

Figure 5-6: UN/CEFACT XML Schema Files

Each of the Root XML Schema Files defined within the given context category namespace always includes the BIE XML Schema file and the BDT XML Schema File. The BIE XML Schema File always includes the BDT XML Schema File. The BDT XML Schema File always includes zero or more BCL XML Schema Files and zero or more BIS XML Schema Files. The BDT XML Schema File also always imports zero or more CCL XML Schema Files and zero or more CIS XML Schema

Files. The Business Code List XML Schema Files may also import a single Common Code List XML Schema File, only if it restricts the list of common codes for the given context category value for the business use case. Dependencies exist among the various files as shown in Figure 5-7.See section 8 XML Schema Files and the corresponding sub-sections.

Each **`xsd:schema`** element used to define an XML Schema Document within an XML Schema File 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 contents of the set of XML Schema within a given namespace are so interrelated that proper management dictates that versioning of all members of the set be synchronized so that incompatible definitions are avoided. All schemas of the set, which are already assigned a single namespace version, are therefore additionally assigned to a single file version number.

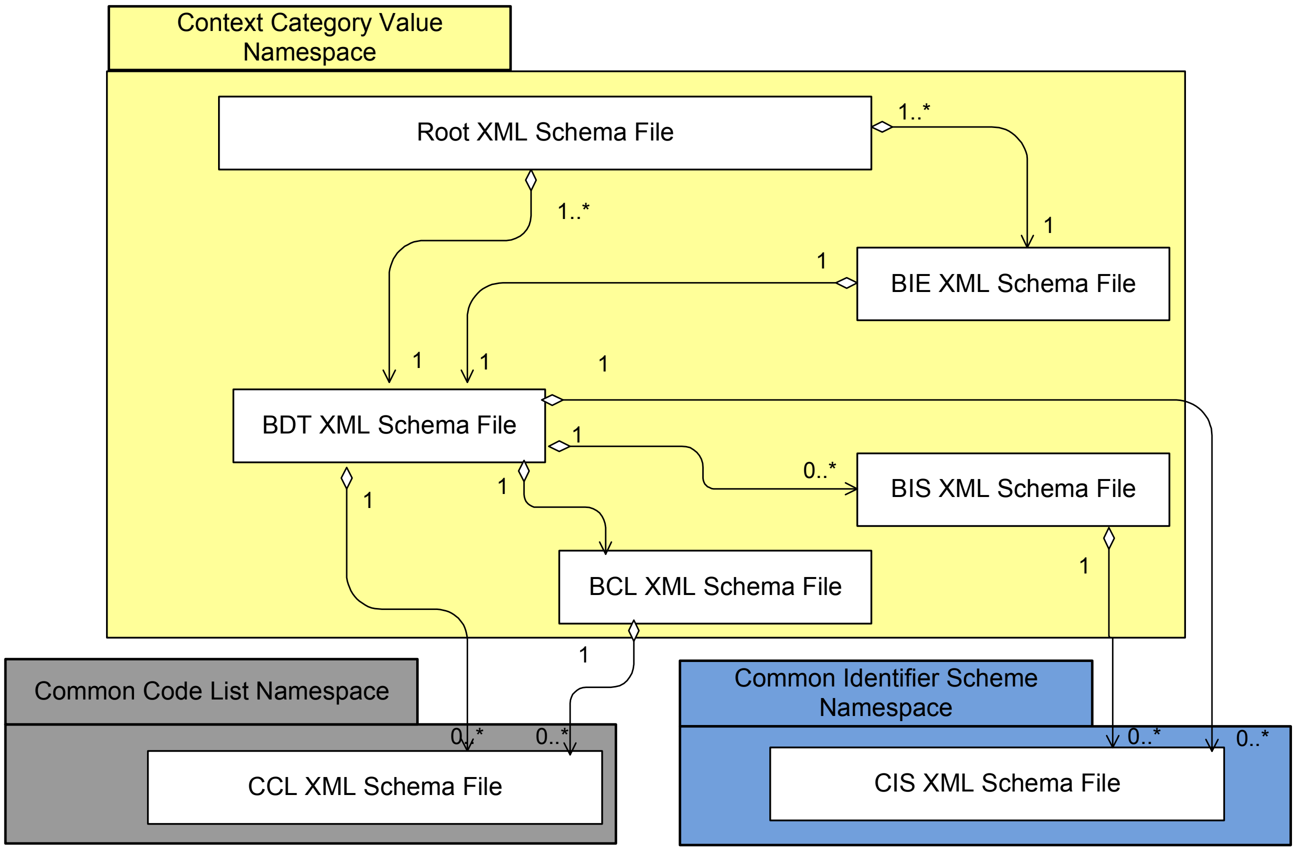


Figure 5-7: UN/CEFACT XML Schema Modularity Scheme

5.7.1 Root XML Schema Files

As expressed in section 5.6 Namespace Scheme, Root XML Schema Files are assigned to a namespace that reflect the dominate context category value of the schema as shown in Figure 5-5. The determination of the dominate context category is at the discretion of the originating organization. The XML Schema File modularity scheme also calls for a set of XML Schema Files that support a Root XML Schema File. This set of XML Schema Files is also assigned to the same dominate context category namespace This approach enables the use of individual context category value focused Root XML Schema Files without importing the entire library. Each Root XML Schema File will define its own dependencies.

There maybe a number of UN/CEFACT Root XML Schema Files, 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 have unique names based on their business function. This business function is defined in the UN/CEFACT Requirements Specification Mapping (RSM) document as the target business information payload.

[R B3E4]	Each Root XML Schema File MUST be named after the **`<BusinessInformationPayload>`** that is expressed in the XML Schema File by using the value of **`<BusinessInformationPayload>`** followed by the words '**`XML Schema File`**' as the name and placing the name in the Header documentation section of the file.	1

As defined in Section 5.3, each root XML Schema File will only contain MAs and ASMAs. The Root XML Schema File will not duplicate reusable XML constructs available in the other XML Schema Files in the same namespace. Instead, the root XML Schema File uses the **`xsd:include`** feature.

[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 Information Entity XML Schema Files

A BIE XML Schema File will be created to define all reusable BIEs within a primary context category value namespace.

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

[R 8238]	A BIE XML Schema File MUST be created within each namespace	1

	that is defined for the primary context category value.	
[R 8252]	The BIE XML Schema Files MUST be named 'Business Information Entity XML Schema File' by placing the name within the Header documentation section of the file.	1

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 XML Schema File including it.

5.7.3 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 or a business scheme or list.

For reference purposes, UN/CEFACT publishes a BDT XML Schema File that consists of all BDTs without restriction to the value domain. This schema file resides in the documentation common namespace and is used for reference purposes only.

[R A2F0]	An unqualified BDT XML Schema File MUST be created in the documentation common namespace to represent the set of unrestricted BDTs.	1

Additional BDT XML Schema Files that contain only the BIEs used in a primary context category namespace will also be published as part of the schema set of that namespace.

[R AA56]	A BDT XML Schema File MUST be created within each namespace that is defined for the primary context category value.	1
[R 847C]	The BDT XML Schema Files MUST be named 'Business Data Type XML Schema File' by placing the name within the header documentation section of the file.	1

5.7.4 Code List XML Schema Files

Code lists published by standards organizations represent a set of commonly accepted codes for use in a variety of business circumstances and contexts. Code lists can be either:

- Unrestricted by an implementation context category values, defined outside of any implementation context category value and expressed as a CCL XML Schema File.
- Defined by an implementation context category value and expressed as a BCL XML Schema File.

Some owning organizations such as UN/CEFACT publish these code lists as an XML Schema File, others do not. The modularity model calls for each code list to be expressed in an XML Schema File. If an external published code list that conforms to the rules of this specification is not already available as an XML Schema File, then a CCL XML Schema File will be created.

[R 8A68]	A Code List XML Schema File MUST be created to convey code list enumerations for each code list being used.	1
[R B0AD]	The name of each Code List XML Schema File as defined in the comment within the XML Schema File MUST be of the form: **`<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>" - Code List XML Schema File"`** Where: • Code List Agency Identifier – Identifies the agency that maintains the code list. • Code List Agency Name – Agency that maintains the code list. • Code List Identification Identifier – Identifies a list of the respective corresponding codes. • Code List Name – The name of the code list as assigned by the agency that maintains the code list.	1

Example 5-12 shows an example of using the CCL Identifiers to name the Code List XML Schema File as described in Rule [R B0AD].

Example 5-12: 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 5-13 shows an example of using the CCL Names to name the Code List XML Schema File as described in Rule [R B0AD].

Example 5-13: Name of UN/CEFACT Security Type Code List XML Schema File Name using Names

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

Additional examples of CCL XML Schema Files can be found at the UN/CEFACT Web site.

5.7.4.1 Common Code List XML Schema Files

A code list is considered common if it is published by a recognized standards organization for use across a broad spectrum of contexts. UN/CEFACT will prepare a CCL for each common code list used by a BDT. Each CCL XML Schema File will contain enumerated values for codes and code values.

[R 942D]	Each CCL XML Schema File MUST contain enumeration values for both the actual codes and the code values.	1

5.7.4.2 Business Code List XML Schema Files

A BCL may be created for a BDT. The BCL can be a restriction or extension to the set of codes in a CCL, be a new code list, or be a union of code lists.. All BCLs are expressed as individual XML Schema Files and are assigned to the same namespace as the XML Schema Files that make use of them. If a BDT that references a BCL is used in different namespaces, then a BDT will be defined and a BCL will be included in each namespace.

Each BCL XML Schema File contains enumerated values for codes and their code values. These enumerated values may be a part of a restriction of a CCL, as a new Code List for the given context category, or as an extension to an existing CCL.

[R A8A6]	Each BCL XML Schema File MUST contain enumeration values for both the actual codes and the code values, through one of the following: • The restriction of an imported CCL. • The extension of a CCL where the codes and values of the CCL are included and the new extensions are added. • The creation of a new Code List that is used within the context category value namespace.	1

5.7.5 Identifier Schemes

Identifier schemes are different than code lists in both concept and functionality. Whereas a code has a value, an identifier is a pointer that is typically devoid of any specific value. Code lists are enumerated lists. Identifier schemes are typically not enumerated.

Identifier schemes will be defined as simple types without enumeration in an Identifier Scheme XML Schema File following the same approach as is used for code lists.

[R AB90]	An Identifier Scheme XML Schema File MUST be created to convey identifier scheme metadata for each scheme being used.	1
[R A154]	The name of each Identifier Scheme XML Schema File as defined in the comment within the XML Schema File MUST be of the form: **`<Identifier Scheme Agency Identifier|Identifier`**	1

	Scheme Agency Name><Identifier Scheme Identification Identifier|Identifier Scheme Name>" Identifier Scheme XML Schema File" Where: • Identifier Scheme Agency Identifier – Identifies the agency that maintains the identifier scheme. • Identifier Scheme Agency Name – Agency that maintains the identifier scheme. • Identifier Scheme Identification Identifier – Identifies the scheme. • Identifier Scheme Name – The name of the identifier scheme as assigned by the agency that maintains the identifier scheme.	

Example 5-14: Name of GS1 Global Trade Item Number Identifier Scheme XML Schema File Name using Identifiers

```
9GTIN - Code List XML Schema File
where:
6 = Agency identifier for GS1 as defined in UN/CEFACT code
    list 3055
GTIN = GS1 Identification identifier for Global Trade Item Number
```

5.7.5.1 Common Identifier Scheme

A common identifier scheme is one that is used for a broad audience in multiple business processes. Common schemes are typically formally published as metadata which fully describe them to enable development of conformant identifiers.

5.7.5.2 Business Identifier Scheme

A business scheme may be defined for a BDT. In cases where some codes in the source CCL are not needed in the business process, the BCL will be a restriction to the CCL. All BCLs are expressed as individual XML Schema Files and are assigned to the same namespace as the XML Schema Files that make use of them. If a BDT that references a BCL is used in different namespaces, then a BDT will be defined and a BCL will be included in each namespace.

[R BD2F]	A Business Identifier Scheme XML Schema File MUST be created for each Business Scheme used by a BDT.	1

Each Business Scheme XML Schema File contains metadata regarding the scheme. If a business scheme is a restriction on a common scheme, the nature of the restriction will be included in the metadata as a business rule in an xsd:annotation xsd:appInfo element.

[R AFEB]	Each Business Identifier Scheme XML Schema File MUST contain metadata that describes the scheme or points to the scheme.	1

5.7.6 Other Standard Bodies BIE XML Schema Files

Other Standards Development Organizations (SDO) 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 [R B998].

In order to achieve interoperability it is critical that these components are consistently represented regardless of which organization they orginate.

[R B564]	Imported XML Schema Files MUST be fully conformant to category 1, 2, 3, 4 and 7 rules as defined in rule [R 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 XML tools attempt to acquire resources using the schema location attribute. The implication of the **`xsi: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 **`xsi: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

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

files). Thus using an absolute path approach with URL-formatted **`xsi:schemaLocation`** values often represents a challenge in practical implementations as it requires open internet connections at 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 within an XML Schema File MUST contain a resolvable relative path URL.	2

Example 5-16: 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 rule [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: **`<xsd:schema ... version=" <major>"p"<minor>["p"<revision>]">`** Where: • **`<major>`** - sequential number of the major version. • **`<minor>`** - sequential number of the minor version • **`<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 whenever 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 optional to mandatory.

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 iteration of a UN/CEFACT XML Schema File there could potentially be a series of minor, or backward compatible, changes. Each minor version will be compatible with both 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.

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.

For a particular namespace, the major version and subsequent minor versions and 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

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

[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

6 Application of Context

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

- Business semantics – The meaning of business information in communication.
    - Meaning can vary between different individuals depending on the context of the sender and the receiver of the information.
    - Meaning can be the same between different individuals depending on the context of the sender and the 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, BIEs represent context specific artifacts for a message. CCTS defines different context categories that capture context category values. BIE artifacts may be defined within any number of combinations of context categories and context category values within a category. BIEs may have the same name with different context values and different content models. As identified in Section 5.6, the namespace mechanism using the primary context category will ensure name collision of similarly named components in different contexts does not occur.

[Note:]

It is possible to extend the namespace described in section 5.6 Namespace Scheme 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 primary context category, all other context category values for every BIE is expressed within the XML Schema definition for each XML Schema Component as an **`xsd:appInfo`** declaration following the structure defined in section 7.5.2 Application Information (AppInfo).

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 and set of rules based on general best practices for those constructs that can be used and to identify those constructs that should not be used to include:

- Overall XML Schema Structure and Rules
- Attribute and Element Declarations
- Type Definitions
- Use of Extension and Restriction
- Annotation

7.1 Overall XML Schema Structure and Rules

7.1.1 XML Schema Declaration

As required by XSD, when defining an XML Schema file the first line indicates the xml version and the encoding it uses. UN/CEFACT XML Schema will use UTF-8 encoding.

[R 88E2]	Every UN/CEFACT XML Schema File MUST use UTF-8 encoding.	1

Example 7-1 shows the declaration of encoding for the XML Schema document.

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 of the schema documentation in the form of **`xsd:comment`** lines will appear. These comments are applicable to the XML Schema file. The template for this is shown in Appendix B in section B.2

[R ABD2]	Every XML Schema File MUST contain a comment that identifies its name immediately following the XML declaration using the format defined in Appendix B-2.	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 is 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. The XML Schema Component will have:

- **elementFormDefault** set to qualified.
- **attributeFormDefault** set to unqualified.
- The prefix **xsd** 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 snippet declaring **schema** component, set the namespace token to **xsd**, set the **elementFormDefault** to qualified and set the **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 defined in 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**.

[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 header comment within the XML Schema File.	1
[R 9443]	The CCTS Metadata XML Schema File MUST reside in its own namespace and be defined in accordance with rule [R 8E2D] and assigned the prefix **ccts**.	1

7.1.5 Constraints on Schema Construction

In addition to general XML Schema structure, constraints on certain XML Schema rules are 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.	46
[R AEBB]	The **`xsd:any`** attribute MUST NOT be used.	46
[R 9859]	Mixed content MUST NOT be used.	1
[R B20F]	**`xsd:redefine`** MUST NOT be used.	46
[R 926D]	**`xsd:substitutionGroup`** MUST NOT be used.	46
[R 8A83]	**`xsd:ID`**/**`xsd:IDREF`** MUST NOT be used.	1

7.2 Attribute and Element Declarations

7.2.1 Attributes

Attributes are only used to convey BDT supplementary components as part of a BDT xsd:type definition. Where the xsd:attributes of an XSD data type definition in XSD part two exist, the BDT will use the xsd data type as its base type and will use the xsd:attributes to represent supplementary components. Where this is not the case, user defined attributes will be declared to represent supplementary components.

[R B221]	Supplementary Components MUST be declared as Attributes.	1
[R AFEE]	User defined attributes MUST only be used for Supplementary Components.	3
[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 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 whose aggregationKind=shared.

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.

7.2.2.2 Empty Elements

In general, the absence of an element in an XML document 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, with a **`xsi:nil`** attribute to indicate that it is intentionally empty.

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

[R B8B6]	Empty elements MUST NOT be used.	3
[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]	Anonymous 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. Examples 7-3 shows a simple type defined in the BDT XML Schema File.

Example 7-3: 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 7-4 shows a simple type defined in a Code List XML Schema File.

Example 7-4: 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 ABIE. 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 meet the business requirements.

[R A4CE]	An **xsd:complexType** MUST be defined for each CCTS ABIE.	1
[R BC3C]	An **xsd:complexType** MUST be defined for each CCTS BDT that cannot be fully expressed using an **xsd:simpleType**.	1

Example 7-5 shows a complex type defined for an Account ABIE.

Example 7-5: 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:sequence and xsd:choice compositors will be used to express the content model for xsd:complexType definitions. The **xsd:all** XML Schema compositor will not be used.

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

7.4 Use of Extension and Restriction

In keeping with CCTS, XML Schema Components are based on the concept that the underlying semantic structures of the BIEs are normative forms of standards that developers are not allowed to alter without coordination with the owner of the component at the data model level. As business requirements dictate, new BIE artifacts will be created in the data model and represented as XML Schema Components by defining new types and declaring new elements. The concept of derivation from existing types through the use of **xsd:extension** and

xsd:restriction will only be used in limited circumstances where their use does not violate this principle.

It is understood that other standards organizations using this specification may choose to use **xsd:extension** and/or **xsd:restriction** to define new constructs that are extended or restricted from existing constructs.

7.4.1 Extension

UN/CEFACT XML Schema Files may only use **xsd:extension** in the BDT XML Schema File to declare attributes to accommodate supplementary components. **xsd:extension** will only be used in an **xsd:complexType** within the BDT XML Schema File, and only for declaring attributes to support supplementary components.

[R AB3F]	**xsd:extension** MUST only be used in the BDT XML Schema File.	4 6
[R 9D6E]	**xsd:extension** MUST only be used for declaring **xsd:attribute**s to accommodate relevant supplementary components.	4 6

Example 7-6 shows an extension of a simple type using the **xsd:extension** mechanism.

Example 7-6: Extension of Simple Type

```
<xsd:complexType name="AmountType">
        <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
        <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                        <xsd:attribute name="unitCode" type="xsd:token"/>
                </xsd:extension>
        </xsd:simpleContent>
</xsd:complexType>
```

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 qualified BDT types that are derived from less qualified or unqualified BDT types.

[R 9947]	**xsd:restriction** MUST only be used in BDT XML Schema Files.	1

Where used, the derived types must always be named uniquely. 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 data type the resulting	1

	type MUST be uniquely named.	

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 the **`xsd:documentation`** and **`xsd:appInfo`** elements within an **`xsd:annotation`** to provide CCTS artifact metadata and context values.

[R 847A]	Each defined or declared construct MUST use the **`xsd:annotation`** element for required CCTS documentation and application information to communicate context.	1

7.5.1 Documentation

The annotation **`xsd:documentation`** will be used to convey the metadata specified by CCTS for CCTS artefacts. Conversely, all elements specified within an **`xsd:documentation`** element will be limited to expressions of CCTS artifact metadata.

The following annotations are required as defined in each of the sub-sections in the section 8 XML Schema Files that correspond to the different CCTS artifacts.

- **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}
- **ObjectClassQualifierName** – Is a word or words which help define and differeniate an ABIE from its associated CC and other BIEs. It enhances the sematic meaning of the DEN to reflect a restriction of the concept, conceptual domain, content model or data value.
- **ObjectClassTermName** – Is a semantically meaningful name for the object class. It is the basis for the DEN.
- **Cardinality** – Indicates the cardinality of the associated artifact.

- **SequencingKey –** Indicates the sequence of the associated artifact within the larger BIE.
- **DictionaryEntryName –** The Data Dictionary Entry Name (DEN) 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.
- **BusinessTermName –** A synonym term under which the artifact is commonly known and used in business. (BusinessTerm)
- **AssociationType** – Indicates the UML Association Kind shared or composition of the ABIE being associated in the ASBIE.
- **PropertyTermName** – Represents a distinguishing characteristic of the object class and shall occur naturally in the definition.
- **PropertyQualifierName** – Is a word or words which help define and differentiate a property. It further enhances the semantic meaning of the property.
- **RepresentationTermName** – An element of the component name which describes the form in which the component is represented.
- **AssociatedObjectClassTermName** – The Associated Object Class Term represented by the artefact.
- **AssociatedObjectClassQualifierTerm** – A term(s) that qualifies the Associated Object Class Term.
- **PrimitiveTypeName –** The name of the primitive type name from the Data Type Catalogue.
- **DataTypeName –** The name of the DataType. This DataType is defined in the Data Type Catalogue.
- **DataTypeQualifierName –** Is a word or words which help define and differentiate a Data Type. It further enhances the semantic meaning of the DataType.
- **DefaultIndicator –** Indicates that the specific Code List Value is the default for the Code List.
- **DefaultValue –** Is the default value.
- **DefaultValueSource –** The source for the default value.
- **SchemeOrListID –** The unique identifier assigned to the scheme or list that uniquely identifies it.
- **SchemeOrListAgencyID –** The unique identifier assigned to the Agency that owns or is responsible for the Scheme or Code List being referenced.
- **SchemeOrListAgencyName –** The name of the Agency that owns or is responsible for the Scheme or Code List being referenced.

- **SchemeOrListModificationAllowed Indicator –** Indicates whether the values being validated can be outside the enumertations specified by the Scheme or Code List.
- **SchemeOrListName –** Name of the Scheme or Code List.
- **SchemeOrListBusinessTermName –** A synonym term under which the Scheme or Code List is commonly known and used in business.

Table 7-1 provides a summary view of the annotation data as defined in this section and the CCTS artifacts in which each is expressed within the resulting XML Schema.

[Note:]

It is important to realize that while this specification lists these artifacts for the documentation there are different types of classes. RSM, ABIE, BBIE, ASBIE and BDT are all Registry Classes in that they are uniquely identifable within the Core Component Library (CCL).

Basic Business Information Entity, Association Business Information Entity, Code List, Code List Value and Supplementary Components are not Registry Classes therefore the do not include the UniqueID or VersionID from the	rsm:RootSchema	ABIE xsd:complexType	BBIE xsd:element	ASBIE: xsd:element	bdt:BusinessDataType	Supplementary Component	Code List	Code List Value
Unique ID	M	M	M	M	M			
Version ID	M	M	M	M	M			
Object Class Qualifier Name	O R	O R						
Object Class Term Name	M	M						
Cardinality			M	M		M		
Sequencing Key			M	M				
Dictionary Entry Name	M	M	M	M	M			
Definition	M	M	M	M	M			
Business Term Name	O R	O R	O R	O R	O R			
Association Type				M				
Property Term Name			M	M	M	M		
Property Qualifier Name			O R	O R				

Representation Term Name			M			M		
Associated Object Class Term Name				M				
Associated Object Class Qualifier Term Name				O R				
Primitive Type Name						M		
Data Type Name					M	M		
Data Type Qualifier Name					M	M		
Default Indicator					M	M		
Default Value					O	O		
Default Value Source					O	O		
Scheme Or List ID					O	O	M	
Scheme Or List Version ID					O	O	O	
Scheme Or List Agency ID					O	O	O	
Scheme Or List Agency Name					O	O	O	
Scheme Or List Modification Allowed Indicator					O	O	M	
Scheme Or List Name					O	O	O	O
Scheme Or List Business Term Name					O R	O R	O R	O R

Key:

M – Mandatory

O – Optional

R – Repeating

Yellow Shading – Not expressed in XML Schema

Table 7-1 Annotation Data Summary

Section 8 XML Schemas and Appendix F specify normative information for the specific annotation required for each of the CCTS artifacts.

This documentation is intended to be used to connect the XML Schema defined artifact to the model artifact in which it is based. This is important for standard XML Schemas and for fully expressed XML Schemas for a runtime implementation.

However, XML Schemas directly used in a runtime implementation may choose not to include this documentation in order to reduce the size of the XML Schema. This is often done in order to increase the throughput of XML Instances and to increase the sheer volume. If this is done the runtime XML Schemas may only be an exact copy of the fully documented XML Schemas with only the annotation documentation (**`xsd:documentation`**) elements removed.

[R A9EB]	Each defined or declared construct MUST use an **`xsd:annotation`** and **`xsd:documentation`** element for required CCTS documentation.	3

As identified in section 7.1.4 CCTS Artifact Metadata, the required elements are declared in the CCTS Metadata XML Schema File. This file will be imported in all Root, BIE, BDT and Code List XML Schema Files in lieu of re-declaring these **`xsd:documentation`** elements.

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

Example 7-8: Example of Annotation Documentation of an ABIE

```
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID>UNBE000000</ccts:UniqueID>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:ObjectClassQualifierName>Customer</ccts:ObjectClassQualifierName>
    <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
    <ccts:DictionaryEntryName>Customer. Account</ccts:DictionaryEntryName>
    <ccts:Definition>The Customer Account.</ccts:Definition>
  </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.

Appendix F section F.1 Annotation Documentation shows the XML Schema definition of annotation documentation for each of the types of components from CCTS.

7.5.2 Application Information (AppInfo)

The annotation **`xsd:appInfo`** will be used to convey the Usage Rules and the Business Context that is applicable for each BIE and BDT artifact and the resulting XML Schema artifacts used to express them.

[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.

Example 7-9 shows the XML Schema definition of the annotation application Information structure **`ccts:UsageRule`**.

Example 7-9: XML Schema definition for annotation appInfo for ccts:UsageRule

```
<xsd:schema
   ...
<xsd:element name="UsageRule" type="ccts:UsageRuleType"/>
   <xsd:complexType name="UsageRuleType">
         <xsd:sequence>
               <xsd:element name="UniqueID" type="bdt:EntityUniqueIdentifierType"/>
               <xsd:element name="Constraint" type="bdt:TextType"/>
               <xsd:element name="ConstraintTypeCode" type="bdt:CodeType"/>
               <xsd:element name="ConditionTypeCode" type="bdt:ConditionTypeCodeType"/>
maxOccurs="unbounded"/>
         </xsd:sequence>
   </xsd:complexType>
   ...
</xsd:schema>
```

Appendix F Section F.2 Annotation Application Information shows the XML Schema definition of the annotation application Information structure for **ccts:BusinessContext**.

Both **ccts:UsageRule** and **ccts:BusinessContext** are applied to each of the XML Schema Components **xsd:element**, **xsd:complexType** and **xsd:simpleType** in order to communicate the usage and context in which the corresponding CCTS artifacts are applicable.

[R 9B07]	Each of the resulting XML Schema Components (**xsd:element**, **xsd:complexType** and **xsd:simpleType**) MUST have an **xsd:annotation xsd:appInfo** declared that includes zero or more **ccts:UsageRule** elements and one or more **ccts:BusinessContext** elements.	1

7.5.2.1 Usage Rules

CCTS defines the concept of usage rules to convey instructions on how to use a CCTS artifact in a given context. 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).

Usage Rules are communicated through zero or more **ccts:UsageRule** XML Schema Elements within an **xsd:appInfo**. Usage rules may be either structured or unstructured. Unstructured usage rule constraint values are expressed as free form text. Structured usage rule constraint values are expressed in a formal constraint language such as the Object Management Group (OMG) Object Constraint Language (OCL).and are suitable for direct application processing.

[R 88DE]	Usage rules MUST be expressed within an **xsd:appInfo ccts:UsageRule** element.	1
[R B851]	The structure of the **ccts:UsageRule** element MUST be: • **ccts:UniqueID** [1..1] – A unique identifier for the UsageRule. • **ccts:Constraint** [1..1] – The actual constraint expression. • **ccts:ConstraintType** [1..1] – The type of constraint E.g.	1

	unstructured, OCL. • **`ccts:ConditionType`** [1..1] – The type of condition. Allowed values are **`pre-condition`**, **`post-condition`**, and **`invariant`**.	

The ccts:ConstraintType value will be taken from a constraint value code list schema.

[R A1CF]	A ccts:ConstraintType code list XML Schema File will be created.	1

7.5.2.2 Business Context

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

The following **`xsd:appInfo`** structures are required as defined in each of the sub-sections in the section 8 XML Schema Files that correspond to the different CCTS artifacts. The BusinessContext defined within each **`xsd:appInfo`** contains one or more **`ccts:ContextUnit`** elements which in turn contains one or more values for each of the identified context categories recognized by CCTS.

- Business Process Context Category
- Business Process Role Context Category
- Supporting Role Context Category
- Industry Classification Context Category
- Product Classification Context Category
- Geopolitical Context Category
- Official Constraints Context Category
- System Capabilities Context Category

[R A538]	Each defined or declared XML Schema artifact MUST use an **`xsd:annotation`** and **`xsd:appInfo`** element to communicate the context of the artifact.	1

Using this structure it is possible to indicate all of the context categories in which a BIE is applicable, and all of the applicable context values within a context category as shown in Example 7-10.

Example 7-10: Use of the `xsd:appInfo` and ccts:BusinessContext

```
<xsd:element name="<name>" type="<type>">
  <xsd:annotation>
  … (documentation) …
  <xsd:appinfo source="urn:un:unece:uncefact:businesscontext….">
         <ccts:UsageRules>
                ...
         </ccts:UsageRules>
         <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>
```

8 XML Schema Files

This section describes how the requirements of the various XML Schema files that are incorporated within the UN/CEFACT library are built through the application of context categories,unique namespaces and the rules of this specification.

- XML Schema Files, Context and Namespaces
- Root XML Schema Files
- Business Information Entities XML Schema Files
- Business Data Type XML Schema Files
- Code List XML Schema Files
    - General Code List XML Schema Components
    - Common Code List XML Schema Components
    - Business Code List XML Schema Components

8.1 XML Schema Files, Context and Namespaces

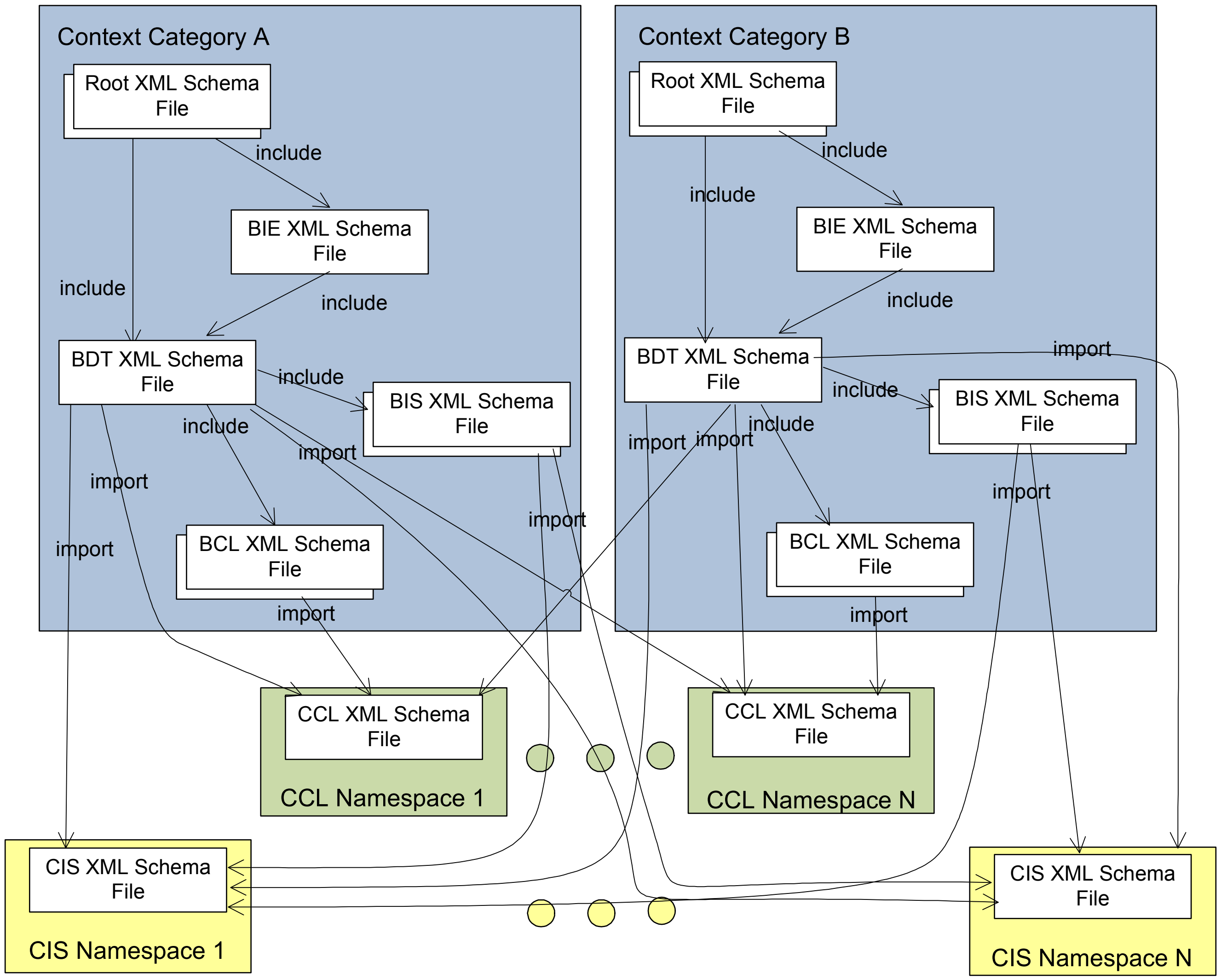
As indicated in section 5.7 XML Schema Files the XML Schema files have dependencies upon one another.

Figure 8-1 further shows these dependencies and shows how these dependencies are realized using the **`xsd:include`** and **`xsd:import`** XML Schema features. Since the primary context category values are implemented within the namespace scheme, all of the XML Schema Files for the given context category value are defined within the corresponding namespace. The XML Schema Files for other values of the context categories are defined in namespaces corresponding to those values.

Figure 8-1 shows two context category values "A" and "B." The namespaces used to express the two context category values are independently declared and may not have any shared dependencies other than Common Code Lists that are independent of all context.

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

[R B96F]	Each Root, BIE, BDT and BCL XML Schema File MUST be assigned to a unique namespace that represents the primary context category value of its contents.	1



1429 **Figure 8-1: Imports and Includes of XML Schema Files for UN/CEFACT**
1430 **Moularity Model**

1431 Example 8-1 shows a namespace declaration for the context category Business
1432 Process Value where the value is Order Management.

1433 **Example 8-1: Namespace for Context Category Business Process – Order Management**

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

1435 Example 8-2 shows how an XML Schema File that is declared within the context
1436 category Business Process Value of Order Management.

1437 **Example 8-2: Schema-element target namespace declaration for context category Business**
1438 **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 use of a semantically meaningful namespace prefix like “**`ordman`**“ for the Business Process – Order Management.

8.2 Root XML Schema Files

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

Figure 8-1 shows multiple Root XML Schema Files defined in two context category based namespaces. Each primary context category value namespace will have 1 to many Root XML Schema Files.

8.2.1 XML Schema Structure

Each Root XML Schema File will be structured in a standardized format as specified in Appendix B 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: Root XML Schema File Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ==================================================================== -->
<!-- =====  [MODULENAME] XML Schema File                            ===== -->
<!-- ==================================================================== -->
<!--
   Schema agency:        UN/CEFACT
   Schema version:       3.0
   Schema date:          18 November 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">
  <!-- ================================================================== -->
  <!-- =====  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>
```

8.2.2 Includes

Every Root XML Schema File in a namespace will include the BIE XML Schema File, and the BDT XML Schema File that reside in that namespace for the specified context category value.

[R B698]	The Root XML Schema File MUST include the BIE and BDT XML Schema Files that reside in its namespace.	1

8.2.3 Root Element Declaration

Each 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 using the XML Schema Component **`xsd:element`**.	1
[R A466]	The name of the root element MUST be the same as the name of the business information payload data dictionary name, with separators and spaces removed.	1
[R 8062]	The root element declaration MUST be defined using an **`xsd:complexType`** that represents the message content contained within the business information payload.	1

Example 8-4 shows an example of Root Element declaration with in a Root XML Schema File.

Example 8-4: Root Element declaration

```
   <!-- ================================================================== -->
   <!-- ===== 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.

8.2.4 Type Definitions

Root XML Schema Files are limited to defining a single MA **`xsd:complexType`** whose content model contains ASMAs that represent the first level BIEs for a business information payload.

[R 8837]	Each Root XML Schema File MUST define a single **`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 Root XML Schema Files complex type definition.

Example 8-5: Root element complex type name

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

8.2.5 Annotations

8.2.5.1 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 (**`xsd:annotation xsd:documentation`**) present in that includes: • UniqueID (mandatory): The identifier that uniquely identifies the business information payload, the root element. • VersionID (mandatory): The unique identifier that identifies the version of the business information payload, the root element. • ObjectClassQualifierName (zero or more): Is a word or words which help define and differeniate an ABIE from its	1

	associated CC and other BIEs. It enhances the sematic meaning of the DEN to reflect a restriction of the concept, conceptual domain, content model or data value. • ObjectClassTermName (mandatory): Is a semantically meaningful name of the Object class. It is the basis for the DEN. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the business information payload. • Definition (mandatory): The semantic meaning of the root element. • BusinessTermName (optional, repeating): A synonym term under which the payload object is known by in industry.	

Example 8-6 shows the definition of the annotation documentation for the Root Element.

Example 8-6: Root element annotation documentation

```
<xsd:group name="RootSchemaDocumentation">
          <xsd:sequence>
                 <xsd:element name="UniqueID"
type="bdt:EntityUniqueIdentifierType"/>
                 <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                 <xsd:element name="ObjectClassQualifierName" type="bdt:NameType"
minOccurs="0" maxOccurs="unbounded"/>
                 <xsd:element name="ObjectClassTermName" type="bdt:NameType"/>
                 <xsd:element name="DictionaryEntryName" 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>
```

8.2.5.2 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). The specific context values for the Root Element represent the context values for the Root XML Schema File.

8.3 Business Information Entity XML Schema Files

A UN/CEFACT BIE XML Schema File contains all of the ABIEs used for the context category value that is reflected in the namespace. This BIE XML Schema File will be used (included into) in all of the UN/CEFACT Root XML Schema Files within the namespace.

8.3.1 Schema Structure

Each BIE XML Schema File will be structured in the standardized format detailed in Appendix B. The specific format is shown in Example 8-7 and must adhere to the format of the relevant sections in Appendix B.

Example 8-7: 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:           18 November 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">
  <!-- =================================================================== -->
  <!-- ===== Includes                                              ===== -->
  <!-- =================================================================== -->
  ... see includes ...
  <!-- =================================================================== -->
  <!-- ===== Type Definitions                                     ===== -->
  <!-- =================================================================== -->
  ... see type defintions ...
</xsd:schema>
```

8.3.2 Includes

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

[R 8FE2]	The BIE XML Schema File MUST contain an **`xsd:include`** statement for the BDT XML Schema File that resides in the same namespace.	1

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

Example 8-8: Include of BDT XML Schema File

```
  <!-- =================================================================== -->
  <!-- ===== Includes                                              ===== -->
  <!-- =================================================================== -->
  <!-- ===== Include of Business Data Type XML Schema File          ===== -->
  <!-- =================================================================== -->
  <xsd:include schemaLocation="BusinessDataType_1p0.xsd"/>
```

8.3.3 Type Definitions

8.3.3.1 ABIE Type Definitions

Every ABIE with the same primary context category is defined as an xsd:complexType in the BIE XML Schema File for that primary context category namespace.

[R AF95]	For every object class (ABIE) identified in a primary context category, a named **`xsd:complexType`** MUST be defined in its corresponding BIE XML Schema File.	1

The name of the xsd:complexType will represent the DEN of the BIE.

[R 9D83]	The name of the ABIE **xsd:complexType** MUST be the **ccts:DictionaryEntryName** with the spaces and separators removed, with approved abbreviations and acronyms applied and with the '**Details**' suffix replaced with '**Type**'.	1

The content model of the **xsd:complexType** will be defined such that it reflects each property of the object class. The content model of the ABIE complex type definitions will include element declarations for BBIEs, element declarations for ASBIEs whose **associationKind=composite**, or element references for ASBIEs whose **associationKind=shared**.

The cardinality and sequencing of each ABIE Property will be determined by the **Cardinality** and **Sequencing Key** values of the source ABIE.

[R 90F9]	The cardinality and sequencing of the elements within an ABIE **xsd:complexType** MUST be as defined by the corresponding ABIE values in the syntax neutral model.	1

In defining the content model, both xsd:sequence and xsd:choice compositors are allowed.

[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

When using the **xsd:sequence** and **xsd:choice** content models in a type definition their order must be carefully managed. An **xsd:sequence** should not contain another **xsd:sequence** directly as there is no additional value. An **xsd:choice** should not contain another **xsd:choice** directly as there is no additional value. However, it is permissible to interweave **xsd:sequence** and **xsd:choice** within a single **xsd:complexType** definition to whatever level of nesting is desired.

[R 81F0]	Repeating series of only **xsd:sequence** MUST NOT occur.	1
[R 8FA2]	Repeating series of only **xsd:choice** MUST NOT occur.	1

Example 8-9 show an example of using **xsd:sequence**.

Example 8-9: Sequence compositor within an ABIE type definition

```
<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="bie: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 8-10 show an example of using **`xsd:choice`**.

Example 8-10: Choice compositor within an ABIE type definition

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

Example 8-11 shows an example of interweaving **`xsd:sequence`** and **`xsd:choice`**.

Example 8-11: Sequence + Choice compositors within an ABIE type definition

```
<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>
```

8.3.3.2 BBIE Type Definitions

BBIEs are defined as local elements and are either of xsd:simpleType or xsd:complexType.

[R A21A]	Every BBIE within the containing ABIE MUST have a named **`xsd:simpleType`** (If the BBIE BDT includes only the content component) or **`xsd:complexType`** (If the BBIE BDT includes one or more supplementary components).	1

The name of the BBIE type will represent the DEN of the BBIE.

[R 8B85]	Every BBIE type MUST be named the property term and qualifiers and the representation term of the basic business information entity (BBIE) it represents with the word '**`Type`**' appended.	1

8.3.3.3 ASBIE Type Definitions

ASBIEs are declared as either local or global elements whose xsd:complexType is that of the xsd:complexType of the associated ABIE it represents. No additional type definition is required.

8.3.4 Element Declarations and References

8.3.4.1 ABIE Element Declarations

Every ABIE will have a globally declared element. This global element reflects the unique DEN of the ABIE within the namespace to which it is assigned and will be of the **`xsd:complexType`** that represents it.

[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

8.3.4.2 BBIE Element Declarations

Every BBIE will have a locally declared element that is part of the content model of the ABIE to which it belongs.

[R 89A6]	For every BBIE identified in an ABIE, a named `xsd:element` MUST be locally declared within the `xsd:complexType` representing that ABIE.	1

The name of the BBIE element will reflect the name of the BBIE devoid of the object class and object class qualifiers.

[R AEFE]	Each BBIE element name declaration MUST be the property term and qualifiers and the representation term of the BBIE.	1

Simplification of the BBIE Property name for the representation terms of `Identification`, `Indicator`, and `Text` are allowed to improve semantic expression.

[R 96D9]	For 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]	For 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

The BBIE element will be of the `xsd:simpleType` or `xsd:complexType` as defined in Section 8.3.3.2.

[R BCD6]	Every BBIE element declaration MUST be of the BusinessDataType that represents the source basic business	1

	information entity (BBIE) data type.	

Example 8-12 shows an Account ABIE complexType declaration that contains BBIE element declarations that make use of the appropriate BDTs.

Example 8-12: BBIE 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="bie: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=”bie:BuyerPartyType/>
        </xsd:sequence>
</xsd:complexType>
```

8.3.4.3 ASBIE Element Declarations

For ASBIEs whose **`ccts:AggregationKind`** value is **`composite`**, a local element for the associated ABIE will be declared in the content model of the associating ABIE **`xsd:complexType`**.

[R 9025]	For every ASBIE whose **`ccts:AggregationKind`** value = **`composite`**, a local element for the associated ABIE MUST be declared in the associating ABIE **`xsd:complexType`** content model.	1

For each ASBIE whose **`ccts:AggregationKind`** value is **`shared`**, a global element is declared. See section 5.5 Reusability Schema earlier this specification.

[R 9241]	For every ASBIE whose **`ccts:AggregationKind`** value = **`shared`**, a global element MUST be declared.	1

The name of the ASBIE local or global element will reflect the name of the ASBIE devoid of the associating object class and qualifiers.

[R A08A]	Each ASBIE 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

The ASBIE local or global element will be of the **`xsd:complexType`** of the associated ABIE.

[R B27C]	Each ASBIE element declaration MUST use the **`xsd:complexType`** that represents its associated ABIE.	1

Example 8-13 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 8-13: ASBIE element declaration and reference within an ABIE type definition

```
<xsd:element name=”InvoiceTradeLineItem” type=”InvoiceTradeLineItemType”/>
<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>
```

8.3.5 Annotation

8.3.5.1 ABIE Complex Type Definition

Every ABIE complexType definition must include structured annotation documentation.

[R ACB9]	For every ABIE **`xsd:complexType`** definition a structured set of annotations MUST be present in the following pattern: • UniqueID (mandatory): The unique identifier that identifies an ABIE instance in a unique and unambiguous way. • VersionID (mandatory): An unique identifier that identifies the version of an ABIE. • ObjectClassQualifierName (optional, repeating): Is a word or ordered words which help define and differeniate the associated ABIE from its CC. • ObjectClassTermName (mandatory): Is a semantically meaningful name of the object class of the ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the ABIE. • Definition (mandatory): The semantic meaning of the ABIE. • BusinessTermName (optional, repeating): A synonym term in which the ABIE is commonly known.	1

In addition, every ABIE **`xsd:complexType`** definition will have structured annotation application information that reflects its context and any usage rules.

[R B0BA]	For every ABIE **`xsd:complexType`** definition a structured set of **`xsd:annotation xsd:appInfo`** elements MUST be present that fully declare its context.	1
[R BCE9]	For every ABIE usage rule, the ABIE **`xsd:complexType`** definition MUST contain a structured set of **`xsd:annotation xsd:appInfo`** elements in the following pattern: • **`ccts:UniqueID`** • **`ccts:Constraint`** • **`ccts:ConstraintType`** • **`ccts:ConditionType`**.	1

Example 8-14 shows the annotation documentation of an ABIE complexType definition.

Example 8-14: ABIE complex type definition annotation

```
  <xsd:complexType name="AccountType" >
    <xsd:annotation>
      <xsd:documentation xml:lang="en-US">
          <ccts:UniqueID>UNBE000000</ccts:UniqueID>
          <ccts:VersionID>0.00</ccts:VersionID>
          <ccts:ObjectClassQualifierName></ccts:ObjectClassQualifierName>
          <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
          <ccts:DictionaryEntryName>Account</ccts:DictionaryEntryName>
          <ccts:Definition>Comminicates the Account information.</ccts:Definition>
          <ccts:BusinessTermName></ccts:BusinessTermName>
      </xsd:documentation>
      <xsd:appInfo>
          As shown in Appendix F
      </xsd:appInfo>
    </xsd:annotation>
</xsd:complexType>
```

8.3.5.1.1 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: • UniqueID (mandatory): The unique identifier that identifies an ABIE instance in a unique and unambiguous way. • VersionID (mandatory): An unique identifier that identifies the version of an ABIE. • ObjectClassQualifierName (optional, repeating): Is a word or ordered words which help define and differeniate the associated ABIE from its CC. • ObjectClassTermName (mandatory): Is a semantically meaningful name of the object class of the ABIE. • DictionaryEntryName (mandatory): The Data Dictionary	1

	Entry Name (DEN) of the ABIE. • Definition (mandatory): The semantic meaning of the ABIE. • BusinessTermName (optional, repeating): A synonym term in which the ABIE is commonly known.	

The global element declaration for ABIEs is used exclusively for referencing by ASMAs. Since multiple ASMAs can reference a single global ABIE element declaration in different contexts with different usage rules, the context and usage rules for global ABIE element declarations can not be explicitly stated in the BIE XML Schema File. However, the context and usage rules can be stated when the global ABIE element is referenced using xsd:ref as part of the content model of the MA.

8.3.5.1.2 BBIE Element

Every BBIE element declaration will include structured annotation documentation.

[R B8BE]	For every BBIE **`xsd:element`** declaration a structured set of **`xsd:annotation`** **`xsd:documentation`** elements MUST be present in the following pattern: • Cardinality (mandatory): Indicates the cardinality of the BBIE within the containing ABIE. • SequencingKey (mandatory): Indicates the sequence of the BBIE within the containing ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the BBIE. • Definition (mandatory): The semantic meaning of the associated BBIE. • BusinessTermName (optional, repeating): A synonym term in which the BBIE is commonly known. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the BBIE. • PropertyQualifierName (optional repeating): Is a word or words which help define and differentiate the BBIE. • RepresentationTermName (mandatory): An element of the component name that describes the form in which the BBIE is represented.	1

In addition, every BBIE will have structured annotation application information that reflects its context and any defined usage rules.

[R 95EB]	For every BBIE **`xsd:element`** declaration a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements MUST be present that fully declare its context.	1
[R 8BF6]	For every BBIE usage rule, the BBIE xsd:element declaration	1

	MUST contain a structured set of **`xsd:annotation xsd:appInfo`** elements in the following pattern: • **`ccts:UniqueID`** • **`ccts:Constraint`** • **`ccts:ConstraintType`** • **`ccts:ConditionType`**.	

Example 8-15 shows the annotation documentation of a BBIE Element.

Example 8-15: BBIE element annotation

```
<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:Cardinality>1</ccts:Cardinality>
         <ccts:SequencingKey>1</ccts:SequencingKey>
         <ccts:DictionaryEntryName>Account. Identificaton.
Identifier</ccts:DictionaryEntryName>
         <ccts:Definition>The Account Identification Identifier.</ccts:Definition>
         <ccts:BusinessTermName></ccts:BusinessTermName>
         <ccts:PropertyTermName></ccts:PropertyTermName>
         <ccts:PropertyQualifierName></ccts:PropertyQualifierName>
         <ccts:RepresentationTermName></ccts:RepresentationTermName>
    </xsd:documentation>
    <xsd:appInfo>
         As shown in Appendix F for context and usage rules
    </xsd:appInfo>
  </xsd:annotation>
</xsd:element>
```

8.3.5.1.3 ASBIE Element

The global element declaration for ASBIEs is used exclusively for referencing by ABIEs. Since multiple ABIEs can reference a single global ASBIE element declaration in different contexts with different usage rules, most of the metadata, context and usage rules for global ASBIE element declarations can not be explicitly stated in the element declaration and the xsd:annotation xsd:documentation elements will be limited.

[R 8D3E]	Every ASBIE global element declaration MUST have a structured set of **`xsd:annotation xsd:documentation`** elements in the following pattern: • AssociationKind (mandatory): Indicates the UML AssociationKind value of **`shared`** or **`composite`** of the associated ABIE. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the ASBIE. • PropertyQualifierName (optional repeating): Is a word or words which help define and differentiate the ASBIE. • AssociatedObjectClassName (Mandatory): The name of the	1

	associated object class. • AssociatedObjectClassQualifierName (optional, repeating): a name or names that qualify the associated object class.	

Context and usage rules can be stated when the global ASBIE element is referenced using xsd:ref as part of the content model of the ABIE. ASBIEs declared locally, and every xsd:ref occurrence of a ASBIE declared globally, will include structured annotation documentation.

Every ASBIE local element declaration or **`xsd:ref`** occurrence in the content model of an ABIE will include structured annotation documentation.

[R 926A]	Every ASBIE **`xsd:element`** declaration or **`xsd:ref`** occurence MUST have a structured set of **`xsd:annotation xsd:documentation`** elements present in the following pattern: • Cardinality (mandatory): Indicates the cardinality of the ASBIE within the containing ABIE. • SequencingKey (mandatory): Indicates the sequence of the ASBIE within the containing ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the ASBIE. • Definition (mandatory): The semantic meaning of the ASBIE. • BusinessTermName (optional, repeating): A synonym term in which the ASBIE is commonly known. • AssociationKind (mandatory): Indicates the UML AssociationKind value of **`shared`** or **`composite`** of the associated ABIE. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the ASBIE. • PropertyQualifierName (optional repeating): Is a word or words which help define and differentiate the ASBIE. • AssociatedObjectClassName (Mandatory): The name of the associated object class. • AssociatedObjectClassQualifierName (optional, repeating): a name or names that qualify the associated object class.	1

In addition, every ASBIE **`xsd:element`** local declaration or **`xsd:ref`** occurrence in the content model of an ABIE will have structured annotation application information that reflects its context and any defined usage rules.

[R 9D87]	Every ASBIE **`xsd:element`** declaration or ASBIE **`xsd:ref`** to an ABIE global element declaration MUST contain a structured set of **`xsd:annotation xsd:appInfo`** elements that fully declare its	1

	context.	
[R A76D]	Every ASBIE usage rule **xsd:element** declaration or ASBIE **xsd:ref** to an ABIE global element declaration MUST contain a structured set of **xsd:annotation xsd:appInfo** elements in the following pattern: • **ccts:UniqueID** • **ccts:Constraint** • **ccts:ConstraintType** • **ccts:ConditionType**.	1

Example 8-16 shows the annotation documentation of an ASBIE Element. In this case the ASBIE is declared as a shared AggregationKind which results in a global element.

Example 8-16: ASBIE global element declaration annotation

```
<xsd:element name="DelayedShipmentDeliveryStatus" type="bie:DeliveryStatusType"
minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
         <ccts:AssociationKind>composite</ccts:AssociationKind>
         <ccts:PropertyTermName>Shipment</ccts:PropertyTermName>
         <ccts:PropertyQualifierName>delayed</ccts:PropertyQualifierName>
         <ccts:AssociatedObjectClassName>Status</ccts:AssociatedObjectClassName>
         <ccts:AssociatedObjectClassQualifier>Delivery</ccts:AssociatedObject
ClassQualifier>
    </xsd:documentation>
</xsd:annotation>
</xsd:element>
```

Example 8-17 shows the annotation documentation of an ASBIE Element. In this case the ASBIE is declared as a composite AggregationKind which results in a local element.

Example 8-17: ASBIE local element declaration annotation

```
<xsd:element name="DelayedShipmentDeliveryStatus" type="bie:StatusType"
minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
         <ccts:Cardinality>1</ccts:Cardinality>
         <ccts:SequencingKey>1</ccts:SequencingKey>
         <ccts:DictionaryEntryName>Order. Delayed_ Shipment. Delivery_
Status</ccts:DictionaryEntryName>
         <ccts:Definition>The delivery status of the delayed shipment for this
order.</ccts:Definition>
         <ccts:BusinessTermName></ccts:BusinessTermName>
         <ccts:AssociationKind>composite</ccts:AssociationKind>
         <ccts:PropertyTermName>Shipment</ccts:PropertyTermName>
         <ccts:PropertyQualifierName>delayed</ccts:PropertyQualifierName>
         <ccts:AssociatedObjectClassName>Status</ccts:AssociatedObjectClassName>
   <ccts:AssociatedObjectClassQualifier>Delivery</ccts:AssociatedObjectClassQualifie
r>
    </xsd:documentation>
    <xsd:appInfo>
         As shown in Appendix F for context and usage rules
    </xsd:appInfo>
</xsd:annotation>
```

```
</xsd:element>
```

Example 8-18 shows the annotation documentation of a reference to an ASBIE Element.

Example 8-18. ASBIE element REF annotation

```
<xsd:element ref="DelayedShipmentDeliveryStatus" minOccurs="0"
maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US">
         <ccts:Cardinality>1</ccts:Cardinality>
         <ccts:SequencingKey>1</ccts:SequencingKey>
         <ccts:DictionaryEntryName>Order. Delayed_ Shipment. Delivery_
Status</ccts:DictionaryEntryName>
         <ccts:Definition>The delivery status of the delayed shipment for this
order.</ccts:Definition>
         <ccts:BusinessTermName></ccts:BusinessTermName>
         <ccts:AssociationKind>shared</ccts:AssociationKind>
         <ccts:PropertyTermName>Shipment</ccts:PropertyTermName>
         <ccts:PropertyQualifierName>delayed</ccts:PropertyQualifierName>
         <ccts:AssociatedObjectClassName>Status</ccts:AssociatedObjectClassName>
  <ccts:AssociatedObjectClassQualifier>Delivery</ccts:AssociatedObjectClassQualifie
r>
    </xsd:documentation>
    <xsd:appInfo>
         As shown in Appendix F for context and usage rules
    </xsd:appInfo>
</xsd:annotation>
</xsd:element>
```

8.4 Business Data Type XML Schema Files

Multiple BDT XML Schema Files are created. One reference BDT XML Schema File will be created that contains all approved BDTs as published in the BDT catalogue. Additional BDT XML Schema Files will be created that define all BDTs used in a primary context category namespace. The BDT XML Schema File names must follow the UN/CEFACT XML Schema File naming approach. .

8.4.1 Use of Business Data Type XML Schema Files

The reference BDT XML Schema File will not be included as part of the modularity model, rather it is used as a reference. The primary context category BDT XML Schema Files will be used by the BIE XML Schema File and all Root Element XML Schema Files defined in the same primary context category namespace.

8.4.2 XML Schema Structure

Each BDT XML Schema File will be structured in a standard format to ensure consistency and ease of use.

The format is shown in Example 8-19. Each BDT XML Schema File must adhere to the format of the relevant sections as detailed in Appendix B.

Example 8-19: BDT XML Schema file structure

```
<?xml version="1.0" encoding="utf-8"?>
<!-- ================================================================== -->
<!-- =====  Business Data Type XML Schema File                    ===== -->
<!-- ================================================================== -->
```

```
<!--
  Schema agency:        UN/CEFACT
    Schema version:     3.0
    Schema date:        18 November 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">
  <!-- =================================================================== -->
  <!-- ===== Includes                                                ===== -->
  <!-- =================================================================== -->
  ... see includes ...
  <!-- =================================================================== -->
  <!-- ===== Imports                                                ===== -->
  <!-- =================================================================== -->
  ... see imports ...
  <!-- =================================================================== -->
  <!-- ===== Type Definitions                                       ===== -->
  <!-- =================================================================== -->
  ... see type definitions ...
</xsd:schema>
```

8.4.3 Imports and Includes

Each BDT XML Schema File will use **xsd:include** to make use of any BCL XML Schema Files and BIS XML Schema Files being used by the BDT XML Schema Components. Each BDT XML Schema File will also use **xsd:import** to make use of any CCL XML Schema Files and CIS XML Schema Files being used by a BDT within the BDT XML Schema File.

[R 8E0D]	The BDT XML Schema File MUST include (**xsd:include**) the BCL XML Schema Files and BIS XML Schema Files that are defined in the same namespace.	1
[R B4C0]	The BDT XML Schema File MUST import (**xsd:import**) the CCL XML Schema Files and CIS XML Schema Files that are used by a BDT contained within the file.	1

8.4.4 Type Definitions

The BDT XML Schema Components are defined as either an **xsd:complexType** or **xsd:simpleType**.

[R AE00]	Each CCTS BDT artifact within the UN/CEFACT Data Type Catalogue used by the Root XML Schema Files and the BIE XML Schema File within a given namespace MUST be defined as an **xsd:simpleType** or **xsd:complexType** in the BDT XML Schema File with the given namespace.	1

As defined in the Data Type Catalogue a BDT content component Business Value Domain (BVD) can contain either a set of primitives or a code list or point to an identifier scheme. This means that a data type can be defined to have one of several primitives or one or more code lists or one or more identifier schemes. When the BDT is defined in the BDT XML Schema File it will be defined to reflect a single primitive, single code list, the list of code list combinations, or a single identifier scheme.

8.4.4.1 Business Value Domain Expressed By Primitives

When a BDT content component Business Value Domain (BVD) is defined by a primitive, and the primitive facets are supported by the facets of an XSD built-in data type, the BDT will be defined as an xsd:simpleType.

[R 9908]	For every BDT whose content component BVD is defined by a primitive whose facets map directly to the facets of an XSD built-in data type, the BDT MUST be defined as a named **`xsd:simpleType`**.	1
[R B91F]	Every BDT whose content component BVD is defined by a primitive whose facets map directly to the facets of an **`xsd:simpleType`** MUST contain one **`xsd:restriction`** element.	1
[R 9910]	The **`xsd:restriction`** element used in a BDT content component BVD defined by a primitive MUST include an **`xsd:base`** attribute that defines the specific XSD built-in data type required for the content component.	1

If a BDT uses a primitive type to express its content component BVD, it is defined with a name that reflects the data type qualifiers and data type term and the primitive type name.

[R A7B8]	The name of a BDT that uses a primitive to define its content component BVD MUST be the BDT **`ccts:DataTypeQualifier(s)`** if any, plus the **`ccts:DataTypeTerm`**, plus the primitive type name, followed by the word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied.	1

Example 8-20 provides three examples of BDT names, where primitives are used.

Example 8-20: BDT Type Definition Names when Primitive is used

```
CodeTokenType
Where Code is the Data Type Term and Token is the primitive.

PercentDecimalType
Where Percent is the Data Type Term and Decimal is the primitive.

AstronomicalUnitFloatType
Where Astronomical Unit is the Data Type Qualifier, Amount is the Data Type Term,
and Float is the primitive.
```

8.4.4.2 Content Component Business Value Domain Expressed By Code List

If a BDT uses a single BCL or CCL to define its content component BVD, it is defined as an **`xsd:simpleType`** that contains an **`xsd:restriction`** element whose **`xsd:base`** attribute is set to the code lists defined **`xsd:simpleType`** (See Section 8.5.1.4).

[R AA60]	A BDT whose content component BVD is defined as an **`xsd:simpleType`** whose base is a single code list MUST contain an **`xsd:restriction`** element with the **`xsd:base`** attribute set to the code lists defined xsd:simpleType.	1

The name of a BDT that uses a single code list directly reflects the data type qualifiers and data type term and a code list suffix.

[R 8DB1]	The name of A BDT that uses a single code list to define its content component BVD MUST be its **`ccts:DataTypeQualifier(s)`** if any, plus the **`ccts:DataTypeTerm`**, plus the code list suffix, followed by the word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied. The code list suffix MUST be the following: (Any repeated words are eliminated.) • **`<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>`** Where. • Code List Agency Identifier – is the identifier for the agency that code list is from. • Code List Agency Name – is the name of the agency that maintains the code list. • Code List Identification Identifier – is the identifier for the given code list. • Code List Name – is the name for the code list.	1

Example 8-21 shows examples of BDT definition names where the code lists used are expressed in the type definition names.

Example 8-21: BDT Type Definition Names

```
Amount54217Type
Where:
5 is the Code List Agency Identifier
4217 is the Code List Identification Identifier
```

Example 8-22 shows a declaration using a code list in a BDT.

Example 8-22: BDT type definition using one code list

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

8.4.4.3 Business Value Domain Expressed By Multiple Code Lists

If a BDT content component BVD is defined as a choice of two or more code lists, it will be defined as a **`xsd:simpleType`** that contains an **`xsd:union`** element whose **`xsd:memberType`** attribute includes the **`xsd:simpleType`** definitions of the code lists to be included.

[R AAD1]	A BDT whose content component BVD is defined by a choice of two or more code lists MUST be defined as an **`xsd:simpleType`** that contains an **`xsd:union`** element whose **`xsd:memberType`** attribute includes the **`xsd:simpleType`** definitions of the code lists to be included.	1

The name of a BDT that uses multiple code lists reflects the data type qualifiers and data type term and a suffix that uniquely points to the unioned code list.

[R 973C]	The name of a BDT that uses multiple code lists MUST be it's **`ccts:DataTypeQualifier(s)`** if any, plus the **`ccts:DataTypeTerm`**, plus the code list suffix, followed by the word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied. The suffix MUST be the following: (Any repeated words are eliminated) • **`<Code List Agency Identifier|Code List Agency Name><Code List Identification`**	1

	Identifier|Code List Name> Where: • Code List Agency Identifier – is the identifier for the agency that code list is from. • Code List Agency Name – is the name of the agency that maintains the code list. • Code List Identification Identifier – is the identifier for the given code list. • Code List Name – is the name for the code list.	

Example 8-23 shows an example of using two code lists in a BDT.

Example 8-23: Combination of Two Code Lists

```
<xsd:simpleType name="AccountDutyCodeclm64437clm65153Type">
        <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
        <xsd:union memberType="clm64437:AccountTypeCodeContentType
                clm65153:DutyTaxFeeTypeCodeContentType"/>
</xsd:simpleType>
```

8.4.4.4 Content Component Business Value Domain Expressed By Identifier Scheme

If a BDT uses an identifier scheme to define its content component BVD, the BDT is defined as an **`xsd:simpleType`** that contains an **`xsd:restriction`** element whose **`xsd:base`** attribute is set to the identifier scheme defined **`xsd:simpleType`** (See Section X.X).

[R A861]	If a BDT content component BVD is defined as an **`xsd:simpleType`** whose base is an identifier scheme, it MUST contain an **`xsd:restriction`** element with the **`xsd:base`** attribute set to the identifier scheme defined **`xsd:simpleType`**.	1

The name of a BDT that uses an identifier scheme to define its content component BVD reflects the data type qualifiers and data type term and an identifier scheme suffix.

[R 8F96]	The name of A BDT that uses an identifier scheme to define its content component BVD MUST be its **`ccts:DataTypeQualifier(s)`** if any, plus the **`ccts:DataTypeTerm`**, plus the identifier scheme suffix, followed by the word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied.. The code list suffix MUST be the following: (Any repeated words are eliminated.) • **`<Identifier Scheme Agency Identifier|Identifier Scheme Agency Name><Identifier Scheme Identification`**	1

	`Identifier|Identifier Scheme Name>` Where. • Identifier Scheme Agency Identifier – is the identifier for the agency that code list is from. • Identifier Scheme Agency Name – is the name for the Agency that owns the identifier scheme. • Identifier Scheme Identification Identifier – is the identifier for the given identifier scheme. • Identifier Scheme Name – is the name for the identifier scheme.	

Example 8-24 shows examples of BDT definition names where the identifier scheme used are expressed in the type definition names.

Example 8-24: BDT Type Definition Names

```
Amount54217Type
Where:
5 is the Code List Agency Identifier
4217 is the Code List Identification Identifier
```

Example 8-25 shows an example of a BDT that uses an Identifier Scheme type.

Example 8-25: Using an Identifier Scheme for a BDT content component BVD

```
  <xsd:simpleType name="SocialSecurityIdentifierType">
         <xsd:annotation>
                ... see annotation ...
         </xsd:annotation>
         <xsd:restriction base="xxxxx ContentType">
                <xsd:length value="9"/>
         </xsd:restriction>
  </xsd:simpleType>
```

8.4.4.5 BDT with Supplementary Components

Supplementary components refine the BDT content component by providing additional information. Every BDT has zero or more Supplementary Components. If a BDT has supplementary components, and those supplementary components do not map directly to the facets of an XSD built-in datatype, the BDT will be defined as an **`xsd:complexType`** with **`xsd:simpleContent`** and an **`xsd:extension`** element whose **`base`** attribute is set to either a primitive type or an identifier scheme or a code list. Each Supplementary Component is expressed as an **`xsd:attribute`** whose **`name`** is set to the DEN of the given Supplementary Component.

[R AB05]	Every BDT that includes one or more Supplementary Components MUST be defined as an **`xsd:complexType`**	1
[R AAA5]	Every BDT **`xsd:complexType`** definition MUST have an **`xsd:simpleContent`** expression whose **`xsd:extension base`**	1

	attribute is set to the primitive type or scheme or list that defines its Content Component Business Value Domain.	
[R 890A]	Every BDT **`xsd:complexType`** definition MUST include an **`xsd:attribute`** declaration for each Supplementary Component.	1
[R ABC1]	The name of the Supplementary Component xsd:attribute must be the DEN of the Supplementary Component with periods, spaces, and other separators removed.	1

The name of a BDT that is defined as an xsd:complexType will be unique and will reflect the primitive or scheme or list that represents its content component business value domain.

[R 90FB]	The name of a BDT that includes one or more Supplementary Components MUST be: • The BDT **`ccts:DataTypeQualifier(s)`** if any, plus • The **`ccts:DataTypeTerm`**, plus • The suffix of the Content Component Business Value Domain where: 　○ The suffix is the primitive type name, the code list token, the series of code list tokens, or the identifier scheme token. plus • The **`ccts:DictionaryEntryName`** for each Supplementary Component present following the order defined in the Data Type Catalogue, plus • The suffix that represents the Supplementary Component BVD where the suffix is the primitive type name, the code list token, the series of code list tokens, or the identifier scheme token, plus • The word '**Type**'. • With all separators removed and approved abbreviations and acronyms applied.	1

Example 8-26 shows an example of a data type with a content component primitive and a Supplementary Component that contains a code list.

Example 8-26: Business Data type with a content component primitive BVD and a Supplementary Component that contains a code list

```
<xsd:complexType name="AmountDecimalCurrencyCodeClm54217Type">
        <xsd:annotation>
                ... see annotation ...
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:decimal">
                <xsd:attribute name="currencyCode" type="clm54217:CurrencyCodeContentType" use="optional">
                    <xsd:annotation>
                    ... see annotation ...
                    </xsd:annotation>
                  </xsd:attribute>
                </xsd:extension>
              </xsd:simpleContent>
  </xsd:complexType>
```

8.4.4.6 Restricted BDTs

BDTs may have either their content component,and/or supplementary component restricted. At the data model level, restrictions can take the form of restrictions to the Business Value Domain (BVD) of the BDT content component or supplementary component. Restictions can also take the form of restrictions to the cardinality of the BDT supplementary component – to include the presence or absence of the supplementary component. Restrictions to the BVD can be in the form of restrictions to the primitive facets or to the scheme or list used to define the value domain.

At the XML level, restrictions can take the form of restrictions to the BDT content component BVD. This is accomplished by creating a new restricted BDT xsd:simpleType derived from the less restricted or unrestricted BDT xsd:simpleType. Restrictions can also take the form of restrictions to the supplementary component BVD. This is accomplished by creating a new restricted BDT **`xsd:complexType`** that is derived from from the less qualified or unqualified BDT **`xsd:complexType`**.

Restrictions can also take the form of restrictions to the BDT content or supplementary component BVD. This is also accomplished by creating a new restricted BDT that is derived from the less restricted or unrestricted BDT **`xsd:complexType`**.

[R 80FD]	Every restricted BDT 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

Non-standard variations are defined as those that are outside the bounds of the normally defined BVD for the underlying BDT. If non-standard variations from the base type are required, these will be defined as an **`xsd:restriction`** derivation from a custom type.

[R A9F6]	Every restricted BDT XML Schema Component **`xsd:type`** definition requiring a non-standard variation from its base type MUST be derived from a custom type.	1

[Note:]

If a non-standard variation of the standard date time built-in data types is required, for example year month, then a BDT of the Core Data Type TextType needs to be defined, with the appropriate restrictions specified, e.g. a pattern, to specify the required format.

Example 8-27 shows a restricted BDT definition.

Example 8-27: Restricted BDT 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>
...
```

8.4.4.6.1 Restrictions to Content Component

Restrictions to the content component result in the creation of a new qualified BDT.

through restriction to the allowed **`ccts:ContentComponent`** and/or **`ccts:SupplementaryComponent`** primitive facets of the unrestricted BDT type definition, or through restrictions to the common code list, business code list, common identifier scheme or business identifier scheme used to define the BVD when those are used in lieu of a primitive.

8.4.4.6.2 Restrictions to Supplementary Component

Restrictions to the supplementary component result in the creation of a new qualified BDT

through restriction to the allowed **`ccts:ContentComponent`** and/or **`ccts:SupplementaryComponent`** primitive facets of the unrestricted BDT type definition, or through restrictions to the common code list, business code list, common identifier scheme or business identifier scheme used to define the BVD when those are used in lieu of a primitive.

8.4.5 Attribute and Element Declarations

There are no element declarations in the BDT XML Schema Files. The only allowed attributes are supplementary components, which are defined locally in the BDT.

[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]	In the BDT XML Schema File, local **`xsd:attribute`** declarations MUST only represent CCTS Supplementary Components for the BDT for which they are declared.	1

8.4.6 Annotations

8.4.6.1 Annotation Documentation

8.4.6.1.1 BDT Types

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

[R BFE5]	Every BDT definition MUST contain a structured set of annotation documentation in the following sequence and pattern: • UniqueID (mandatory): The unique identifier that identifies the BDT in a unique and unambiguous way. • VersionID (mandatory): An unique identifier that identifies the version of the BDT. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the BDT. • Definition (mandatory): The semantic meaning of the BDT. • BusinessTermName (optional, repeating): A synonym term in which the BDT is commonly known. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the BDT and shall occur naturally in the definition. • DataTypeName (mandatory): The name of the DataType. The possible values for the DataType are defined in the Data Type Catalogue. • DataTypeQualifierName (mandatory): Is a word or words which help define and differentiate a Data Type. It further enhances the semantic meaning of the DataType. • DefaultIndicator (mandatory): Indicates that the specific Code List Value is the default for the Code List.	1

	• DefaultValue (optional): Is the default value. • DefaultValueSource (optional): Indicates the source for the default value. • SchemeOrListID (optional): The unique identifier assigned to the scheme or list that uniquely identifies it. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the Scheme or Code List being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the Scheme or Code List being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the enumertations specified by the Scheme or Code List. • SchemeOrListName (optional): Name of the Scheme or Code List. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the Scheme or Code List is commonly known and used in business. (BusinessTerm)	

Example 8-28 shows the annotation documentation structure declaration for each BDT.

Example 8-28: BDT annotation documentation definition

```
  <xsd:group name="BDTDocumentation">
          <xsd:sequence>
                 <xsd:element name="UniqueID"
type="bdt:EntityUniqueIdentifierType"/>
                 <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                 <xsd:element name="DictionaryEntryName" type="bdt:NameType"/>
                 <xsd:element name="Definition" type="bdt:TextType"/>
                 <xsd:element name="BusinessTermName" minOccurs="0"
maxOccurs="unbounded"/>
                 <xsd:element name="PropertyTermName" type="bdt:NameType"/>
                 <xsd:element name="DataTypeName" type="bdt:NameType"/>
                 <xsd:element name="DataTypeQualifierName" type="bdt:NameType"/>
                 <xsd:element name="DefaultIndicator" type="bdt:IndicatorType"/>
                 <xsd:element name="DefaultValue" type="bdt:TextType"
minOccurs="0"/>
                 <xsd:element name="DefaultValueSource" type="bdt:TextType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListID" type="bdt:IDType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType" minOccurs="0"/>
                 <xsd:element name="SchemeOrListName" type="bdt:NameType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrLisBusinessTermtName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
```

Example 8-29 shows an example annotation documentation of a BDT.

Example 8-29: BDT annotation element

```
... see type definition ...
<xsd:annotation>
          <ccts:UniqueID>UNDT000000-000</ccts:UniqueID>
          <ccts:VersionID>0.00</ccts:VersionID>
          <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
          <ccts:Definition></ccts:Definition>
          <ccts:DataTypeName></ccts:DataTypeName>
          <ccts:DataTypeQualifierName></ccts:DataTypeQualifierName>
          <ccts:DefaultIndicator>true</ccts:DefaultIndicator>
          <ccts:DefaultValue></ccts:DefaultValue>
          <ccts:DefaultValueSource></ccts:DefaultValueSource>
          <ccts:SchemeOrListID></ccts:SchemeOrListID>
          <ccts:SchemeOrListVersionID></ccts:SchemeOrListVersionID>
          <ccts:SchemeOrListAgencyID></ccts:SchemeOrListAgencyID>
          <ccts:SchemeOrListAgencyName></ccts:SchemeOrListAgencyName>
<ccts:SchemeOrListModificationAllowedIndicator><ccts:SchemeOrListModificationAllowe
dIndicator>
          <ccts:SchemeOrListName></ccts:SchemeOrListName>
          <ccts:SchemeOrListBusinessTermName></ccts:SchemeOrListBusinessTermName>
  </xsd:documentation>
</xsd:annotation>
... see type definition ...
```

8.4.6.1.2 BDT Type Supplementary Components

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

[R 9C95]	Every supplementary component **`xsd:attribute`** declaration MUST contain a structured set of annotation documentation MUST in the following pattern: • Cardinality (mandatory): Indicates the cardinality of the SC within the containing BDT. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the SC and shall occur naturally in the definition. • RepresentationTermName (mandatory): An element of the component name that describes the form in which the SC is represented. • PrimitiveTypeName (mandatory): The name of the SC PrimitiveType. • DataTypeName (mandatory): The name of the DataType. The possible values for the DataType are defined in the Data Type Catalogue. • DataTypeQualifierName (mandatory): A word or words which help define and differentiate a Data Type. It further enhances the semantic meaning of the DataType. • DefaultIndicator (mandatory): Indicates that the specific Code List Value is the default for the Code List or identifier scheme.	1

	• DefaultValue (optional): Is the default value. • DefaultValueSource (optional): Indicates the source for the default value. • SchemeOrListID (optional): The unique identifier assigned to the scheme or list that uniquely identifies it. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the identifier scheme or code list being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the identifier scheme or code list being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the enumertations specified by the identifier scheme or code list. • SchemeOrListName (optional): Name of the identifier scheme or code list. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the identifier scheme or code list is commonly known and used in business. (BusinessTerm)	

Example 8-30 shows the annotation documentation definition for each BDT SC.

Example 8-30: BDT SC annotation documentation definition

```
   <xsd:group name="BDTSCDocumentation">
          <xsd:sequence>
                 <xsd:element name="Cardinality" type="bdt:NumericType"/>
                 <xsd:element name="PropertyTermName" type="bdt:NameType"/>
                 <xsd:element name="RepresentationTermName" type="bdt:NameType"/>
                 <xsd:element name="PrimitiveTypeName" type="bdt:NameType"/>
                 <xsd:element name="DataTypeName" type="bdt:NameType"/>
                 <xsd:element name="DataTypeQualifierName" type="bdt:NameType"/>
                 <xsd:element name="DefaultIndicator" type="bdt:IndicatorType"/>
                 <xsd:element name="DefaultValue" type="bdt:TextType"
minOccurs="0"/>
                 <xsd:element name="DefaultValueSource" type="bdt:TextType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListID" type="bdt:IDType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType" minOccurs="0"/>
                 <xsd:element name="SchemeOrListName" type="bdt:NameType"
minOccurs="0"/>
                 <xsd:element name="SchemeOrLisBusinessTermtName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
   </xsd:group>
```

8.4.6.2 Annotation Application Information (AppInfo)

The annotation **`xsd:appInfo`** is expressed for all BDT artifacts defined in BDT XML Schema Files. The UsageRules and the context is communicated as defined in section 7.5.2, Application Information (AppInfo). All UsageRules and contexts in which the BDT is applicable is expressed in the **`xsd:appInfo`**.

8.5 Code List XML Schema Files

Codes are an integral component of any 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 XML Instance documents to be fully validated by 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 Common Code Lists. For example, many of the code lists that exist in the United Nations Code List (UNCL) will be stored as Common Code List XML Schema Files for use within other UN/CEFACT XML Schema Files.

[R 9E40]	Each code list used by a BDT or BBIE MUST be defined in its own XML Schema File.	2

UN/CEFACT recognizes two basic types of code lists:

- Common Code List (CCL) – Universally defined for use in all contexts. Generally maintained by UN/CEFACT and other standards bodies.
- Business Code List (BCL) which are defined within a given context of their use. They may be defined as:
    - A subset of an existing CCL or
    - Additions to an existing CCL or
    - A new Code List that is needed within the context of use for a given context category namespace

8.5.1 General Code List XML Schema Components

Both Common Code List XML Schema Files and Business Code List XML Schema Files define codes using a consistent approach.

8.5.1.1 Code List XML Schema File Structure

Each Code List XML Schema File will be structured in a standard format in order to ensure consistency and ease of use. This structure is show in Example 8-31.

Example 8-31: Code List XML Schema File structure

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ==================================================================== -->
<!-- =====   6Recommendation20 - Code List XML Schema File           ===== -->
```

```
<!-- ==================================================================== -->
<!--
   Schema agency:        UN/CEFACT
   Schema version:       2.0
   Schema date:          16 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>
```

8.5.1.2 Code List XML Schema Name

The name of Code List XML Schema Files are dependent upon the agency that defines them and the name of the code list itself.

[R 849E]	Code List XML Schema File names MUST be of the form: **`<Agency Identifier	Agency Name>_<List Identification Identifier	List Name>_<Version Identifier>.xsd`** All periods, spaces, or other separators are removed except for the "." before xsd and the "_" between the names. Where: • 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. • Agency Name – the name of the agency that maintains the list. • List Identification Identifier – identifies a list of the respective corresponding codes or ids. • List Name – the name of a list of codes. • Version Identifier – identifies the version.	2

8.5.1.3 Element Declarations

A Code List XML Schema File contains one global element declaration. This global element is a unique identifier for the code list and is mandatory for UN/CEFACT Code List XML Schema Files. Other organizations using this specification may choose to not provide the Code List Root Element and still be in compliance with this specification.

[R 8D1D]	Each Code List XML Schema File MUST declare a single global element.	3

The global element serves as the root element and is of the one xsd:simpleType that is defined in the Code List XML Schema File.

[R BE84]	The Code List XML Schema File global element MUST be of the **xsd:simpleType** that is defined in the Code List XML Schema File.	3

Example 8-32 shows a root element declaration for a code list.

Example 8-32: Code list global root element declaration

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

The actual implementation of the code list is through the use of its **xsd:simpleType** by a BDT BVD or BBIE.

8.5.1.4 Type Definitions

Each Code List XML Schema File will have one named **xsd:simpleType** defined. The name of this type will correspond to the code list name with the word '**ContentType**' appended.

[R A8EF]	Each Code List XML Schema File MUST define one, and only one, named **xsd:simpleType** for the content component.	1
[R 92DA]	The Code List XML Schema File **xsd:simpleType** name MUST be the name of the code list root element with the word '**ContentType**' appended.	1

Code List contents are expressesed using **xsd:enumeration**, where each value of the code list is defined using **xsd:value**.

[R 962C]	Each code in a Code List XML Schema File MUST be expressed as **xsd:enumeration**, where the **xsd:value** for the enumeration is the actual code value.	1

Example 8-33 shows a simple type definition used in a code list.

Example 8-33: Code list xsd:simpleType definition

```
<!-- =================================================================== -->
<!-- ===== 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>
```

8.5.1.5 Annotation

8.5.1.5.1 Annotation Documentation

8.5.1.5.1.1 Code List Documentation

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

[R A142]	Every Code List MUST contain a structured set of annotation documentation in the following sequence and pattern: • SchemeOrListID (mandatory): The unique identifier assigned to the code list. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the code list being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the code list being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the enumerations specified by the code list. • SchemeOrListName (optional): Name of the code list. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the code list is commonly known and used in business. (BusinessTerm)	1

Example 8-34 shows the declaration of the code list documentation structure.

Example 8-34: Code list documentation structure

```
  <xsd:group name="CodeListDocumentation">
          <xsd:sequence>
                  <xsd:element name="SchemeOrListID" type="bdt:IDType"/>
                  <xsd:element name="SchemeOrListVersionID" type="bdt:IDType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType"/>
```

```
                 <xsd:element name="SchemeOrListBusinessTermName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
```

8.5.1.5.1.2 Code List Value Documentation

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

[R A814]	Each code list **`xsd:enumeration`** MUST contain a structured set of annotations in the following sequence and pattern: • Name (mandatory): The name of the code. • Description (optional): Descriptive information concerning the code.	1

Example 8-35 shows the annotation documentation definition for the enumerations values of a code list.

Example 8-35: Code list enumeration annotation documentation

```
  <xsd:simpleType name="PaymentMethodCodeContentType">
          <xsd:restriction base="xsd:token">
                 <xsd:enumeration value="1"> Name (mandatory): The name of the
code.
Description (optional): Descriptive information concerning the code.

                        <xsd:annotation>
                               <xsd:documentation xml:lang="en">
                                      <ccts:Name>Direct payment</ccts:Name>
                                      <ccts:Description>An assigned invoice has
been paid by the buyer to the factor.</ccts:Description>
                               </xsd:documentation>
                        </xsd:annotation>
                 </xsd:enumeration>
          </xsd:restriction>
  </xsd:simpleType>
```

8.5.2 Common Code List XML Schema Components

CCL's are universally defined for all contexts and maintained by standards bodies. CCL XML Schema Files will be imported into the context specific namespaces that use them.

8.5.2.1 Namespace Name for Common Code Lists

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

[R 992A]	Code list XML Schema File namespaces MUST use the following pattern:	1

URN:	**`urn:<organization>:<org hierarchy>*[:<org hierarchy level n>]:codelist:common:<major>:<status>:<name>`**
URL:	**`http://<organization>/<org hierarchy>*[/<org hierarchy level n>]/codelist/common/<major>/<status>/<name>`**

Where:

- organization – 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.
- codelist – A fixed value token for common codelists.
- common – A fixed value token for common codelists.
- major – The Major version number of the codelist.
- status – The status of the schema as: draft|standard
- name – The name of the XML Schema File (using upper camel case) with periods, spaces, or other separators and the words 'schema module' removed.
    - Code list names are further defined as: <Code List Agency Identifier|Code List Agency Name> ><divider><Code List Identification Identifier|Code List Name>

      Where:

        - Code List Agency Identifier – is the identifier for the agency that code list is from.
        - Code List Agency Name – is the name of the agency that maintains the code list.
        - Divider – the divider character for URN is ':' the divider character for URL is '/'.
        - Code List Identification Identifer – is the identifier for the given code list.
        - Code List Name – is the name for the code list.

Example 8-36 shows a namespace name of a code list using an agency and a code list identifier at draft status.

Example 8-36: Code list namespace name 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 8-37 shows a namespace name of a proprietary code list at draft status.

Example 8-37: Propreitary code list namespace name 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 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
```

Example 8-38 shows a namespace name of a code list with and agency and code list identifier at standard status.

Example 8-38: Code list namespace name 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 8-39 shows a namespace name of a proprietary code list with a status of standard.

Example 8-39: 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 of code lists published by external organisations is outside of the control of UN/CEFACT, UN/CEFACT published code lists expressed in XML Schema Files will follow the rules expressed in this specification.

8.5.2.2 XML Schema Namespace Token for Common Code Lists

A unique token will be defined for each namespace for common code lists. The token is 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.

The agency maintaining the code list will be identified either 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 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.

[R 9FD1]	Each UN/CEFACT maintained CCL XML Schema File MUST be represented by a unique token constructed as follows: **`clm<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>`** Such that any repeated words are eliminated. Where: • Code List Agency Identifier – is the identifier for the agency that code list is from. • Code List Agency Name – is the name of the agency that maintains the code list. • Code List Identification Identifier – is the identifier for the given code list. • Code List Name – is the name for the code list.	2

Example 8-40 shows a code list token with an agency and code list identifier.

Example 8-40: 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 8-41 shows a code list token for a business data type with an agency and code list identifiers.

Example 8-41: Code list token for a qualified BDT 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 8-42 shows a code list token for a proprietary code list.

Example 8-42: 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 8-43.

Example 8-43: 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">
```

[Note:]

Developers are encouraged to follow the above rules when customizing XML Schema for code lists to ensure that there are no namespace conflicts.

8.5.2.3 Imports and Includes

UN/CEFACT CCL XML Schema Files are standalone XML Schema Files and will not import or include any other XML Schema Files.

[R 86C8]	CCL XML Schema Files MUST NOT import or include any other XML Schema Files.	1

8.5.2.4 Type Definitions

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

[R B40B]	Each CCL XML Schema File **`xsd:simpleType`** MUST use an **`xsd:restriction`** element whose base attribute is **`xsd:token`**.	1

Example 8-44 shows the simple type definition for a code list.

Example 8-44: CCL xsd:simpleType definition

```
  <xsd:simpleType name="PaymentMethodCodeContentType">
        <xsd:restriction base="xsd:token">
              <xsd:enumeration value="1">
                    <xsd:annotation>
                          See annotation
                    </xsd:annotation>
              </xsd:enumeration>
        </xsd:restriction>
</xsd:simpleType>...
```

8.5.2.5 Annotation

8.5.2.5.1 Annotation Documentation

CCL XML Schema documentation follows the same structure as defined in section 8.5.1.4.1 Annotation Documentation of this specification.

8.5.2.5.2 Annotation Application Information (AppInfo)

Common code lists are applicable to all contexts and therefore do not have context specified within an **`xsd:appInfo`** element.

8.5.3 Business Code List XML Schema Components

Business code lists are Code List XML Schema Files that contain codes that are applicable within the context category for the namespace where it is defined. A BCL XML Schema file maybe used where an existing CCL XML Schema File needs to be extended, where no suitable CCL XML Schema exists, or where the context in which the code list is to be used only needs to make use of a subset of a CCL. This is accomplished by:

- A combination of several individual code lists using **`xsd:union`**,
- A new code list that is applicable for the context, or
- Sub setting an existing code list using **`xsd:restriction`**.

[R 8F2D]	BCL XML Schema file MUST be used to • Extend existing CCL or • Define a codelist where one does not exist or • Restrict the value of a CCL for a context category	1

8.5.3.1 Namespace Name for Business Code Lists

BCLs use the namespace name for the context category in which it is defined. This is described earlier in this specification in section 5.6 Namespace Scheme.

8.5.3.2 UN/CEFACT XML Schema Namespace Token for Business Code Lists

BCL use the namespace token for the context category in which it is defined. This is described earlier in this specification in section 5.6.2 Namespace Tokens. In cases where the BCL is a restricted set of values of a published CCL, the BCL 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 CCL XML Schema File.

8.5.3.3 Imports and Includes

BCL Schema Files may import CCL XML Schema File(s) if the BCL restricts the CCL Schema File content or unions multiple CCL content to create a new BCL.

[R 87A9]	BCL XML Schema Files MUST import only CCL XML Schema Files it uses directly.	1

8.5.3.4 Type Definitions

Each BCL XML Schema file will have a single **`xsd:simpleType`** defined. This type definition will have a **`xsd:restriction`** expression whose base is an XML Schema built-in data type or the **`ContentType`** (s) of the CCL the BCL is using. The **`xsd:restriction`** will be used to convey the content component enumeration value(s).

[R 882D]	In each BCL XML Schema File the **`xsd:restriction`** element base attribute value MUST be set to **`xsd:token`**.or the '**`ContentType`**' from the CCL that is being used.	1

8.5.3.5 Annotation

8.5.3.5.1 Annotation Documentation

BCL XML Schema documentation is the same as CCL XML Schema documentation described in Section 8.5.1.4.1 Annotation Documentation.

8.5.3.5.2 Annotation Application Information (AppInfo)

BCL usage rules and context information is as defined in section 7.5.2, Application Information (AppInfo).

8.6 Identifier Scheme XML Schema Files

Identifiers are an integral component of managing business objects. Identifiers have been developed over time to provide for uniquely identifying one object from another. When identifiers are part of an XML based business information exchange, any identifiers used within the XML document need to be able to be validated by the XML parser as to the identifiers adherence to the scheme that defines it.

Many international, national and sectorial agencies create and maintain identifier schemes. If required to be used within an information flow, these schemes will be defined in their own XML Schema File.

[R A1EE]	Each identifier scheme used by a BDT or BBIE MUST be defined in its own XML Schema file.	2

UN/CEFACT recognizes two basic types of identifier schemes:

- Common Identifier Scheme (CIS) – Universally defined for use in all contexts. Generally maintained by UN/CEFACT and other standards bodies.
- Business Identifier Scheme (BIS) These are identifiers that are defined within a given context of their use. The may be defined as:
    - A restriction on the pattern or allowed values of an existing CIS
    - An extension on the pattern or allowed values of an existing CIS
    - A new CIS that is needed within the context of use for a given context category namespace

8.6.1 General Identifier Scheme XML Schema Components

Both Common Identifier Scheme XML Schema Files and Business Identifier Scheme XML Schema Files define the schemes using a consistent approach.

8.6.1.1 Identifier Scheme XML Schema File Structure

Each Identifier Scheme XML Schema File will be structured in a standard format inorder to ensure consistency and ease of use. This structure is show in Example 8-45.

Example 8-45: Identifier scheme XML Schema File structure

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ==================================================================== -->
<!-- =====  Global Trade Identification Number – Identifier Scheme XML Schema
File===== -->
<!-- ==================================================================== -->
<!--
   Schema agency:        GS1
   Schema version:       1.0
   Schema date:          21 December 2008

   Identifier Scheme name:      Global Trade Identification Number
   Identification Scheme agency:        GS1
   Identification Scheme version:       1

  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">
  <!-- ================================================================== -->
  <!-- ===== Root Element                                           ===== -->
  <!-- =================================================================== -->
         ... see root element declaration ...
  <!-- ================================================================== -->
  <!-- ===== Type Definitions                                       ===== -->
  <!-- =================================================================== -->
  <!--= Type Definition: Global Trade Identification Number Content Type  ==-->
  <!-- =================================================================== -->
         ... see type definition ...
</xsd:schema>
```

8.6.1.2 Identifier Scheme XML Schema Name

The name of Identifier Scheme XML Schema Files are dependent upon the agency that defines them and the name of the identifier scheme itself.

[R A50B]	Identifier Scheme XML Schema File names MUST be of the form: **`<Agency Identifier	Agency Name>_<Scheme Identification Identifier	Scheme Name>_<Version Identifier>.xsd`** All periods, spaces, or other separators are removed except for the "." before xsd and the "_" between the names. Where: • Agency Identifier – identifies the agency that manages the	2

	identifier scheme. The default agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used. • Agency Name – the name of the agency that maintains the scheme. • Scheme Identification Identifier – identifies the identifier scheme. • Scheme Name – the name of the identifier scheme. • Version Identifier – identifies the version of the scheme.	

8.6.1.3 Element Declarations

An Identifier Scheme XML Schema File contains one global element declaration. This global element is a unique identifier for the identifier scheme and is mandatory for UN/CEFACT Identifier Scheme XML Schema Files. Other organizations using this specification may choose to not provide the Identifier Scheme Root Element and still be in compliance with this specification.

[R BFEB]	Each Identifier Scheme XML Schema File MUST declare a single global element.	3

The global element serves as the root element and is of the one xsd:simpleType that is defined in the Identifier Scheme XML Schema File.

[R B236]	The Identifier Scheme XML Schema File root element MUST be of the **`xsd:simpleType`** that is defined in the Identifier Scheme XML Schema File.	3

Example 8-46 shows a root element declaration for an identifier scheme.

Example 8-46: Identifier scheme root element declaration

```
<!-- ================================================================== -->
<!-- ===== Root Element                                           ===== -->
<!-- ================================================================== -->
<xsd:element name="GlobalTradeIdentificationNumber"
type="ism8GTIN:GlobalTradeIdentificationNumberType"/>
```

The actual implementation of the identifier scheme is through the use of its **`xsd:simpleType`** by a BDT BVD or BBIE.

8.6.1.4 Type Definitions

Each Identifier XML Schema File will have one named **`xsd:simpleType`** defined. The name of this type will correspond to the identifier scheme name with the word '**`ContentType`**' appended.

[R 9451]	Each Identifier Scheme XML Schema File MUST define one, and only one, named **`xsd:simpleType`** for the content component.	1

[R 92DA]	The Identifier Scheme XML Schema File **`xsd:simpleType`** name MUST be the name of the identifier scheme root element with the word '**`ContentType`**' appended.	1

The identifiers created by an identifier scheme are never enumerated as shown in Example 8-47.

Example 8-47: Identifier scheme xsd:simpleType name

```
<!-- ===================================================================== -->
<!-- ===== Root Element                                              ===== -->
<!-- ===================================================================== -->
<xsd:element name="GlobalTradeIdentificationNumber"
type="ism8GTIN:GlobalTradeIdentificationNumberType"/>
<!-- ===================================================================== -->
  <!-- ===== Type Definitions                                              ===== -->
  <!-- ===================================================================== -->
  <!-- ==  Type Definition: Global Trade Identification Number Identifier= -->
  <!-- ===================================================================== -->
  <xsd:simpleType name="GlobalTradeIdentificationNumberContentType">
         See type definition
  </xsd:simpleType>
```

8.6.1.5 Annotation

8.6.1.5.1 Annotation Documentation

8.6.1.5.1.1 Identifier Scheme Documentation

Every Identifier Scheme XML Schema file must include structured annotation documentation.

[R B30A]	Every Identifier Scheme MUST contain a structured set of annotation documentation in the following sequence and pattern: • SchemeOrListID (mandatory): The unique identifier assigned to the Identifier Scheme. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the identifier scheme being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the identifier scheme being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the pattern specified by the scheme. • SchemeOrListName (optional): Name of the identifier scheme. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the identifier scheme is commonly known and used in business. (BusinessTerm)	1

Example 8-48 shows the declaration of the annotation documentation for each Identifier Scheme.

Example 8-48: Identifier scheme documentation structure

```
  <xsd:group name="CodeListDocumentation">
         <xsd:sequence>
                <xsd:element name="SchemeOrListID" type="bdt:IDType"/>
                <xsd:element name="SchemeOrListVersionID" type="bdt:IDType"
minOccurs="0"/>
                <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"
minOccurs="0"/>
                <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                <xsd:element name="SchemeOrListName" type="bdt:NameType"
minOccurs="0"/>
                <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType"/>
                <xsd:element name="SchemeOrListBusinessTermName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
  </xsd:group>
```

8.6.2 Common Identifier Scheme XML Schema Components

CIS are universally defined for all contexts and maintained by standards bodies. CIS XML Schema Files will be imported into the context specific namespaces that use them.

8.6.2.1 Namespace Name for Common Identifier Scheme

The namespace name for a CIS is somewhat unique in order to convey some of the supplementary components rather than including them as attributes. Specifically, the namespace structure for an identifier scheme extends the earlier rules for namespace names to include the identifier scheme name in the namespace.

[R 9CCF]	Identifier scheme XML Schema File namespaces MUST use the following pattern: **URN:** **`urn:<organization>:<org hierarchy>*[:<org hierarchy level n>]:identifierscheme:common:<major>:<status>:<name>`** **URL:** **`http://<organization>/<org hierarchy>*[/<org hierarchy level n>]/identifierscheme/common/<major>/<status>/<name>`** Where: • organization – 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	1

organization providing the standard.

- identifierscheme – A fixed value token for common identifier schemes.
- common – A fixed value token for common identifier schemes.
- major – The Major version number of the identifier scheme.
- status – The status of the schema as: draft|standard
- name – The name of the XML Schema File (using upper camel case) with periods, spaces, or other separators and the words ‘schema module’ removed.
    - Identifier scheme names are further defined as: <Identifier Scheme Agency Identifier|Identifier Scheme Agency Name> ><divider><Identifier Scheme Identification Identifier|Identifier Scheme Name>

      Where:

        - Identifier Scheme Agency Identifier – is the identifier for the agency that identifier scheme is from.
        - Identifier Scheme Agency Name – is the name of the agency that maintains the identifier scheme.
        - Divider – the divider character for URN is ‘:’ the divider character for URL is ‘/’.
        - Identifier Scheme Identification Identifer – is the identifier for the given identifier scheme.
        - Identifier Scheme Name – is the name for the identifier scheme.

Example 8-49 shows an identifier scheme namespace where the status of the identifier scheme is in draft status.

Example 8-49: Identifier scheme namespace name with an agency and a identifer scheme identifier at draft status

```
"urn:un:unece:uncefact:identifierscheme:common:D.04A:draft:8:GTIN: "
where
D.04A = the version of the UN/CEFACT directory
8 = the value for GS1 in UN/CEFACT data element 3055 representing
    the Identifier Scheme. Agency. Identifier
GTIN = GS1 data element tag for Global Trade Identification Number representing
    the Identifier Scheme. Identification. Identifier
```

While the versioning of identifier schemes published by external organisations is outside of the control of UN/CEFACT, UN/CEFACT published code lists expressed in XML Schema Files will follow the rules expressed in this specification.

8.6.2.2 XML Schema Namespace Token for Common Identifier Schemes

A unique token will be defined for each namespace for common identifier schemes. The token is constructed based on the identifier of the agency maintaining the identifier scheme and the identifier of the specific identifier scheme as issued by the maintenance agency – except where there is no identifier. When there is no identifier, the name for the agency and/or identifier scheme should be used instead. This will typically be true when proprietary identifier schemes are used. This method of token construction will provide uniqueness with a reasonably short token.

The agency maintaining the identifier scheme will be identified either 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 identifier scheme will 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 identifier scheme will be used.

[R B2BC]	Each UN/CEFACT maintained CIS XML Schema File MUST be represented by a unique token constructed as follows: **`clm<Identifier Scheme Agency Identifier|Identifier Scheme Agency Name><Identifier Scheme Identification Identifier|Identifier Scheme Name>`** Such that any repeated words are eliminated. Where: • Identifier Scheme Agency Identifier – is the identifier for the agency that the identifier scheme is from. • Identifier Scheme Agency Name – is the name of the agency that maintains the identifier scheme. • Identifier Scheme Identification Identifier – is the identifier for the given identifier scheme. • Identifier Scheme Name – is the name for the identifier scheme.	2

Example 8-50 shows an identifier scheme token.

Example 8-50: Identifier scheme token with an agency and an identifier scheme identifier

```
The identifier scheme token for Global Trade Identification Number Identier is
ism8gtin
where
8 = the value for GS1 in UN/CEFACT data element 3055 representing
    the Identifier Scheme. Agency. Identifier
gtin = GS1 data element tag for Global Trade Identification Number representing
    the Identifier Scheme. Identification. Identifier
="unqualified">
```

[Note:]

Developers are encouraged to follow the above rules when customizing XML Schema for code lists to ensure that there are no namespace conflicts.

8.6.2.3 Imports and Includes

UN/CEFACT CIS XML Schema Files are standalone XML Schema Files and will not import or include any other XML Schema Files.

[R A6C0]	CIS XML Schema Files MUST NOT import or include any other XML Schema Files.	1

8.6.2.4 Type Definitions

Each CIS XML Schema file will have a single **`xsd:simpleType`** defined. This type definition will have an **`xsd:restriction`** expression whose base is an XML Schema built-in data type of **`xsd:token`**.

[R 9DDA]	Each CIS XML Schema File xsd:simpleType MUST use an **`xsd:restriction`** element whose base attribute value = **`xsd:token`**.	1

Example 8-51 shows an CIS simpleType definition.

Example 8-51: CIS xsd:simpleType definition

```
<xsd:simpleType name="GlobalTradeIdentificationNumberContentType">
        <xsd:restriction base="xsd:token"/>
</xsd:simpleType>
```

A CIS XML Schema File is only identifying the metadata about the identifier scheme, it is not defining the actual scheme itself since that information is publicly available.

8.6.2.5 Annotation

8.6.2.5.1 Annotation Documentation

CIS XML Schema documentation follows the same structure as defined in section 8.6.1.4.1 Annotation Documentation of this specification.

8.6.2.5.2 Annotation Application Information (AppInfo)

Common identifier schemes are applicable to all context and therefore do not have context specified within **`xsd:appInfo`**.

8.6.3 Business Identifier Scheme XML Schema Components

Business identifier schemes are Identifier Scheme XML Schema Files that define a scheme that is applicable within a context category namespace. A BIS XML Schema file may be used where an existing CIS XML Schema identifier scheme needs to be modified, or where no suitable CIS XML Schema exists. In all cases this is accomplished by creating a new identifier scheme. The BIS will:

- Define a new CIS that is needed within the context of use for a given context category namespace
- Redefine an existing CIS by defining:
    - a restriction on the pattern or allowed values of an existing CIS
    - An extension on the pattern or allowed values of an existing CIS

[R A1E3]	BIS XML Schema file MUST be used to • Define an identifier scheme where one does not exist or • Redefine an existing CIS	1

8.6.3.1 Namespace Name for Business Information Scheme

A BIS uses the namespace name for the context category in which it is defined. This is described earlier in this specification in section 5.6 Namespace Scheme.

8.6.3.2 UN/CEFACT XML Schema Namespace Token for Business Information Scheme

A BIS uses the namespace token for the context category in which it is defined. This is described earlier in this specification in section 5.6.2 Namespace Tokens.

8.6.3.3 Imports and Includes

BIS XML Schema Files do not import or include other XML Schema Files.

[R A4BF]	BIS XML Schema Files MUST NOT use **`xsd:import`** or **`xsd:include`**.	1

8.6.3.4 Type Definitions

Each BIS XML Schema file will have a single **`xsd:simpleType`** defined. This type definition will have a **`xsd:restriction`** expression whose base is an XML Schema built-in data type of **`xsd:token`**. The **`xsd:restriction xsd:token`** facets may be used to define the actual identifier scheme as part of the type definition.

[R 96B0]	Each CIS XML Schema File xsd:simpleType MUST use an **`xsd:restriction`** element whose base attribute value is **`xsd:token`**.	1

Example 8-52 shows a BIS simpleType definition.

Example 8-52: BIS xsd:simpleType definition

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

```
        <xsd:restriction base="xsd:token">
</xsd:simpleType>
```

8.6.3.5 Annotation

8.6.3.5.1 Annotation Documentation

BIS XML Schema documentation is the same as CIS XML Schema documentation described in section 8.5.2.4.1 Annotation Documentation.

8.6.3.5.2 Annotation Application Information (AppInfo)

BIS usage rules and context information is as defined in section 7.5.2, Application Information (AppInfo).

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. An 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

9.5 Supplementary Components

Code lists and identifier schemes can be defined for a business value domain either at model design time or at instance run time. When the code list or identifier scheme is defined at model design time, it is included as part of the BDT definition in the BDT XML Schema File. If a code list or identifier scheme is defined at instance run time, the supplementary component attributes are used to identify the list or scheme. To maximize interoperability and minimize human intervention required at runtime, the preferred approach is to define the scheme or list at model design time. Only in very rare circumstances should the supplementary component attributes for identifying a scheme or list be used.

[R A884]	The attributes for scheme or list supplementary components SHOULD NOT be used within an XML Instance.	1

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:                18 November 2008




  Copyright (C) UN/CEFACT (2008). 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:documentation:common:3:draft"
xmlns:rsm=" urn:un:unece:uncefact:documentation:common:3:draft"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:com=" urn:un:unece:uncefact:documentation:common:3:draft"
urn:un:unece:uncefact:codelist:common:2001:standard:5:4217
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 Business Code List XML Schema Files if used

All schemaLocations are relative from the XML Schema File that is referencing. For the purposes of this appendix we are assuming the references are from a Root Schema File within the same namespace as the includes.

Example B-4: Includes

```
<!-- ================================================================== -->
<!-- =====  Include                                              ===== -->
<!-- ================================================================== -->
<!-- =====  Inclusion of context category BIE XML Schema File     ===== -->
<!-- ================================================================== -->
  <xsd:include schemaLocation="BusinessInformationEntity_3p0.xsd"/>
<!-- ================================================================== -->
<!-- =====  Inclusion of context category BDT XML Schema File     ===== -->
<!-- ================================================================== -->
  <xsd:include schemaLocation=”BusinessDataType_3p0.xsd"/>
<!-- ================================================================== -->
<!-- Inclusion of context specific Business Code List XML Schema File = -->
<!-- ================================================================== -->
  <xsd:include schemaLocation=”BusinessCodeList_1p0.xsd”/>
```

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:2001:standard:5:4217"
schemaLocation="../../../../codelist/common/2001/standard/ISO_CurrencyCode_2001.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:VersionID>3.0</ccts:VersionID>
                 <ccts:ObjectClassTermName>Invoice</ccts:ObjectClassTermName>
                 <ccts:DictionaryEntryName>Invoice</ccts:DictionaryEntryName>
                 <ccts:Definition>Document used to communicate the Invoice for a
Purchase.</ccts:Definition>
             </xsd:documentation>
          </xsd:annotation>
  </xsd:element>
```

Example B-8: Global elements

```
<!-- =================================================================== -->
<!-- ===== Global element                                          ===== -->
<!-- =================================================================== -->
 <xsd:element name="BuyerParty" type="bie:BuyerPartyType"/>
          <xsd:annotation>
             <xsd:documentation>
                 <ccts:UniqueID>UNM0000002</ccts:UniqueID>
                 <ccts:VersionID>3.0</ccts:VersionID>
          <ccts:ObjectClassQualifierName>Party</ccts:ObjectClassQualifierName>
                 <ccts:ObjectClassTermName>Party</ccts:ObjectClassTermName>
                 <ccts:DictionaryEntryName>Buyer. Party</ccts:DictionaryEntryName>
                 <ccts:Definition>The Party that initiated the a
Purchase.</ccts:Definition>
            </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="bie: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="bie: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="bie: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="bie: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 Component

BBIE – Basic Business Information Entity

BCC – Basic Core Component

BDT – Business Data Type

BIE – Business Information Entity

CC – Core Component

ID – Identifier

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ====================================================================== -->
<!-- =====  XMLNDR Dcoumentation Schema File
===== -->
<!-- ====================================================================== -->
<!--
  Schema agency:         UN/CEFACT
  Schema version:        3.0
  Schema date:           18 November 2008

  Copyright (C) UN/CEFACT (2008). 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 xmlns:ccts="urn:un:unece:uncefact:documentation:common:3:standard"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:clm54217="urn:un:unece:uncefact:codelist:common:2001:standard:5:4217"
xmlns:bdt="urn:un:unece:uncefact:data:common:3:standard"
targetNamespace="urn:un:unece:uncefact:documentation:common:3:standard"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- ================================================================== -->
  <!-- ===== Imports

===== -->
  <!-- ================================================================== -->
  <!-- ===== Import of Common Business Data Type
                                      ===== -->
  <!-- ================================================================== -->
  <xsd:import namespace="urn:un:unece:uncefact:data:common:3:standard"
schemaLocation="../../../../data/common/3/standard/BusinessDataType_3p0.xsd"/>
  <!-- ================================================================== -->
  <!-- ===== Import of Code Lists
                                                                      ===== -->
  <!-- ================================================================== -->
  <xsd:import
namespace="urn:un:unece:uncefact:codelist:common:2001:standard:5:4217"
schemaLocation="../../../../codelist/common/2001/standard/ISO_CurrencyCode_2001.xsd
"/>
  <!---->
  <!---->
```

F.1 Annotation Documentation

```

  <xsd:group name="RootSchemaDocumentation">
          <xsd:sequence>
                  <xsd:element name="UniqueID"
type="bdt:EntityUniqueIdentifierType"/>
                  <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
```

```
                <xsd:element name="ObjectClassQualifierName" type="bdt:NameType" 
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="ObjectClassTermName" type="bdt:NameType"/>
                <xsd:element name="DictionaryEntryName" 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>
    <xsd:group name="ABIEDocumentation">
            <xsd:sequence>
                <xsd:element name="UniqueID" 
type="bdt:EntityUniqueIdentifierType"/>
                <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                <xsd:element name="ObjectClassQualifierName" type="bdt:NameType" 
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="ObjectClassTermName" type="bdt:NameType"/>
                <xsd:element name="Cardinality" type="bdt:NumericType"/>
                <xsd:element name="SequencingKey" type="bdt:NumericType"/>
                <xsd:element name="DictionaryEntryName" 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>
    <xsd:group name="BBIEDocumentation">
            <xsd:sequence>
                <xsd:element name="UniqueID" 
type="bdt:EntityUniqueIdentifierType"/>
                <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                <xsd:element name="Cardinality" type="bdt:NumericType"/>
                <xsd:element name="SequencingKey" type="bdt:NumericType"/>
                <xsd:element name="DictionaryEntryName" type="bdt:NameType"/>
                <xsd:element name="Definition" type="bdt:TextType"/>
                <xsd:element name="BusinessTermName" type="bdt:NameType" 
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="PropertyTermName" type="bdt:NameType"/>
                <xsd:element name="PropertyQualifierName" type="bdt:NameType" 
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="RepresentationTermName" type="bdt:NameType"/>
            </xsd:sequence>
    </xsd:group>
    <xsd:group name="ASBIEDocumentation">
            <xsd:sequence>
                <xsd:element name="UniqueID" 
type="bdt:EntityUniqueIdentifierType"/>
                <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                <xsd:element name="Cardinality" type="bdt:NumericType"/>
                <xsd:element name="SequencingKey" type="bdt:TextType"/>
                <xsd:element name="DictionaryEntryName" type="bdt:NameType"/>
                <xsd:element name="Definition" type="bdt:TextType"/>
                <xsd:element name="BusinessTermName" type="bdt:NameType" 
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="AssociationType" 
type="bdt:AssociationTypeCodeType"/>
                <xsd:element name="PropertyTermName" type="bdt:NameType"/>
                <xsd:element name="PropertyQualifierName" type="bdt:NameType" 
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="RepresentationTermName" type="bdt:NameType"/>
            </xsd:sequence>
    </xsd:group>
    <xsd:group name="BDTDocumentation">
            <xsd:sequence>
                <xsd:element name="VersionID" type="bdt:VersionIdentifierType"/>
                <xsd:element name="UniqueID" 
type="bdt:EntityUniqueIdentifierType"/>
                <xsd:element name="DictionaryEntryName" type="bdt:NameType"/>
                <xsd:element name="Definition" type="bdt:TextType"/>
                <xsd:element name="BusinessTermName" minOccurs="0" 
maxOccurs="unbounded"/>
                <xsd:element name="PropertyTermName" type="bdt:NameType"/>
                <xsd:element name="DataTypeName" type="bdt:NameType"/>
                <xsd:element name="DataTypeQualifierName" type="bdt:NameType"/>
                <xsd:element name="DefaultIndicator" type="bdt:IndicatorType"/>
                <xsd:element name="DefaultValue" type="bdt:TextType" 
minOccurs="0"/>
```

```
                  <xsd:element name="DefaultValueSource" type="bdt:TextType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListID" type="bdt:IDType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType" minOccurs="0"/>
                  <xsd:element name="SchemeOrListName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrLisBusinessTermtName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
  <xsd:group name="BDTSCDocumentation">
          <xsd:sequence>
                  <xsd:element name="Cardinality" type="bdt:NumericType"/>
                  <xsd:element name="PropertyTermName" type="bdt:NameType"/>
                  <xsd:element name="RepresentationTermName" type="bdt:NameType"/>
                  <xsd:element name="PrimitiveTypeName" type="bdt:NameType"/>
                  <xsd:element name="DataTypeName" type="bdt:NameType"/>
                  <xsd:element name="DataTypeQualifierName" type="bdt:NameType"/>
                  <xsd:element name="DefaultIndicator" type="bdt:IndicatorType"/>
                  <xsd:element name="DefaultValue" type="bdt:TextType"
minOccurs="0"/>
                  <xsd:element name="DefaultValueSource" type="bdt:TextType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListID" type="bdt:IDType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType" minOccurs="0"/>
                  <xsd:element name="SchemeOrListName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrLisBusinessTermtName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
  <xsd:group name="CodeListDocumentation">
          <xsd:sequence>
                  <xsd:element name="SchemeOrListID" type="bdt:IDType"/>
                  <xsd:element name="SchemeOrListVersionID" type="bdt:IDType"/>
                  <xsd:element name="SchemeOrListAgencyID" type="bdt:IDType"/>
                  <xsd:element name="SchemeOrListAgencyName" type="bdt:NameType"
minOccurs="0"/>
                  <xsd:element name="SchemeOrListModificationAllowedIndicator"
type="bdt:IndicatorType" minOccurs="0"/>
                  <xsd:element name="SchemeOrListName" type="bdt:NameType"/>
                  <xsd:element name="SchemeOrListBusinessTermName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
  <xsd:group name="CodeValueDocumentation">
          <xsd:sequence>
                  <xsd:element name="SchemeOrListName" type="bdt:NameType"/>
                  <xsd:element name="SchemeOrListBusinessTermName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
  <xsd:group name="IdentifierSchemeDocumentation">
          <xsd:sequence>
                  <xsd:element name="SchemeOrListName" type="bdt:NameType"/>
                  <xsd:element name="SchemeOrListBusinessTermName"
type="bdt:NameType" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:group>
  <!---->
  <!---->
```

F.2 Annotation Application Information

```
  <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>
```

```xml
                  <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:element name="UsageRule" type="ccts:UsageRuleType"/>
  <xsd:complexType name="UsageRuleType">
          <xsd:sequence>
                  <xsd:element name="UniqueID"
type="bdt:EntityUniqueIdentifierType"/>
                  <xsd:element name="Constraint" type="bdt:TextType"/>
                  <xsd:element name="ConstraintTypeCode" type="bdt:CodeType"/>
                  <xsd:element name="ConditionTypeCode"
type="bdt:ConditionTypeCodeType"/>
                  <xsd:element name="Name" type="bdt:NameType" minOccurs="0"/>
                  <xsd:element name="BusinessTerm" type="bdt:TextType" minOccurs="0"
maxOccurs="unbounded"/>
          </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

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

Appendix H. Use Cases for 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. This specification support flexible use of code lists. This appendix details the mechanisms for this use.

The five alternative uses for code lists are:

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

Example H-1 is a code snippet from an XML Schema File that uses each of these.

Example H-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"/>
  <!-- Message 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:AmountDecimalType"/>
                <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 BDT XML Schema File defined for the given context category (business process value which is order management).

    - The two specific data types CurrencyCodeType and CalculationCurrencyCodeType are defined as Business Code List that are included through the BDT XML Schema File.
- The BIE XML Schema File defined for the given context category.

The **`xsd:complexType`** named “ProductType” includes five local elements. Each of these elements represents one of the five different code list options.

H.1 Referencing a Common Code List as a Supplementary Component in a Business Data Types

In Example H-1, the element TotalAmount is declared as shown in Example H-2.

Example H-2: Declaration of TotalAmount Element

```
<xsd:element name="TotalAmount" type="ordman:AmountDecimalclm5ISO42173AType"/>
```

As shown in the element declaration, TotalAmount is of the generic CCT AmountType that is implemented in the the context category using the primitive decimal and the CCL ISO code list 42173A resulting in the BDT AmountDecimalclm5ISO42173AType which has been defined in the BDT XML Schema File. The AmountDecimalclm5ISO42173A Type declaration is as show in Example H-3.

Example H-3: Declaration of AmountDecimal 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="AmountDecimalclm5ISO42173AType">
        <xsd:simpleContent>
              <xsd:extension base="xsd:decimal">
                    <xsd:attribute name="currencyCode"
type="clm5ISO42173A:ISO3AlphaCurrencyCodeContentType" use="optional"/>

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

The 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 CCL XML Schema File for the ISO Currency Codes is shown in H-4.

Example H-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>
                       ... see the section for Code Value Documentation ...
                              </xsd:documentation>
                       </xsd:annotation>
                </xsd:enumeration>
                <xsd:enumeration value="AFN">
                       <xsd:annotation>
                              <xsd:documentation>
                       ... see the section for Code Value Documentation ...
                              </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. Only code values from this code list are allowed in a CEFACT conformant instance documents. The resulting instance documents conveyance currency code values are represented as:

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

[Note:]

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

H.2 Referencing any code list using BDT CodeType

The second element in our example message – TaxCurrencyCode – is of the BDT 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 as shown in Figure H-5

Example H-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="listAgencyID" type="xsd:token" use="optional"/>
           <xsd:attribute name="listVersionID" type="xsd:token" use="optional"/>
        </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

When the **`bdt:CodeType`** is used, either the listID indicates the Code List identification. The listAgencyID is the Agency identification that made the code list available. The listVersionID indicates the verision of the code list.

The association to the specific values must be made at runtime. In an instance document this element could be represented as:

```
<TaxCurrencyCode listID="ISO 4217" listVersionID="2001"
listAgencyID="5>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.

H.3 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:CurrencyCodeclm54217-A Type"/>
```

The **`bdt:CurrencyCodeType`** would be defined in the BDT XML Schema File as:

```
<xsd:simpleType name="CurrencyCodeclm54217-AType">
        <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 used is carried in the instance document as this is already defined in the XML Schema.

H.4 Choosing or Combining Values from Several Code Lists

The fourth option is to combine values from diverse code lists by using the **`xsd:union`** element.

The **`xsd:union`** code list approach enables multiple code lists to be used for a single element or attribute. The element declaration in the XML Schema, the element **`CalculationCurrencyCode`** is based on the namespace specific BCL type defined in the context category specific namespace BCL XML Schema File where the **`ordman:CalculationCurrencyCodeclm54217-Nclm54217-AType`** is declared.

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

The **`ordman:CalculationCurrencyCodeclm54217-Nclm54217-AType`** is defined in the BCL XML Schema File with in the context category namespace for Order Management, using an **`xsd:union`** element that unions the code lists together.

```
<xsd:simpleType name="CalculationCurrencyCodeclm54217-Nclm54217-AType">
  <xsd:union memberTypes="clm54217-N:CurrencyCodeContentType
         clm54217-A:CurrencyCodeContentType"/>
</xsd:simpleType>
```

This allows values to come from either the **`clm54217-N:CurrencyCodeContentType`** or from the **`clm54217-A:CurrencyCodeContentType`**. The CCL XML Schema File for **`clm54217-A:CurrencyCodeContentType`** is the same as the one used earlier in this Appendix. The CCL XML Schema File for **`clm54217-N:CurrencyCodeContentType`** is the same as the one used earlier in this Appendix.

The **`xsd:union`** allows the use of code values from different pre-defined code lists in instance documents. The code lists must be imported once in the BCL XML Schema File. The specific code list will be represented by the namespace prefixes (**`clm54217-A`** or **`clm54217-N`**), the element in the instance document will not have the specific code list tokens 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, in this case, **`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`** is that attributes can also make use of these code lists.

[Note:]

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

H.5 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 BCL XML Schema File within the specific context category namespace of the XML Schema Files that use it. This BCL XML Schema File simply contains the restricted set of values used by the context category.

This is accomplished by importing the CCL XML Schema File and using **`xsd:restriction`** to restrict the values to the set of values required. For more please section 8.5.3.4 Type Definitions.

Appendix I. Alternative Business Message Syntax Binding

UN/CEFACT will create the XML syntax binding of its CCTS conformant BIE data models directly from the associations and hierarchies expressed in the Business Message Template for each business message exchange. This approach is based on traditional nesting of all components of the data model.

The XML Schema Specification also supports an alternative to nesting. This alternative, using schema identity constraints (`xsd:key/xsd:unique/xsd:keyRef`), enables referencing and reuse of a given element in instance documents. UN/CEFACT is currently evaluating this alternative for future use to include a method for application at the data model level. In anticipation that the data model issues will be resolved, UN/CEFACT has already developed a set of rules for its XML implementation. These rules and the supporting narrative are presented in this Appendix. Organizations using this Alternative Method will still be considered conformant to this specification, if they adhere to all other conformance requirements and use the rules defined in this Appendix.

I.1 XML Schema Architecture

I.1.1 Message Assembly Considerations

If referencing between specific ABIE's is required in the scope of the root Message Assembly (MA) or of a lower level ABIE, the Business Message Template must specify the list of ABIE's that are implemented as referenced rather than nested properties. This will allow the identity constraints to be generated in the message schema.

I.1.2. Requirements for XML Element Referencing

I.1.2.1 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. The ClaimNotify message shown below, taken from the Insurance Industry, illustrate this.
In Example K-1 and Example K-2 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 reduce redundancy in the message, it is possible to use XML referencing to relate one Person instance to the Policy and Claim instances as an alternate method to nesting information about Person within Policy and Claim.

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.

Example I-1: XML Instance of ClaimNotify using nesting

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

Example I-2: XML Instance of ClaimNotify using referencing

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

I.1.2.2 Other Usages of XML Referencing

Another requirement for XML element referencing is *Dynamic Referencing*.

The requirement is that 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.

I.1.2.3 Schema Validation Requirements for XML References

I.1.2.3.1 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.

Due to its more robust identity constraints, this specification mandates **`key/keyRef`** as the XML referencing technique to be used instead of **`Id/IdRef`**. See sections 7.1.5 Constraints on Schema Construction, I.2.1.1 Constraints on Schema Construction and I.3.1.1 Declaration of the Referencing Constraints.

Referencing between ABIEs occur in the boundaries of a particular ‘scope element’ in the XML document. The scope element is the container of all the elements that can be involved in the identity constraints. These identity constraints act as follows:

- The uniqueness (xsd:unique) or key (xsd:key) constraints define the keys and enforce that a value is unique within the scope element.

The key reference (xsd:keyRef) constraints define the key references and enforce that a value corresponds to a value represented by a uniqueness (xsd:unique) or key (xsd:key) constraint.

Most often the scope 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 scope element.

I.1.2.3.2 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.

I.2 General XML Schema Language Conventions

I.2.1 Overall XML Schema Structure and Rules

I.2.1.1 Constraints on Schema Construction

The XML Schema **`xsd:key, xsd:keyref`** or **`xsd:unique`** identity constraints have the following characteristics that make them preferable to the **`xsd:ID/xsd:IDREF`** technique.

- The keys and relationships between objects are strongly typed. They are declared explicitly in the schema. Each relationship is distinctly defined and specifies exactly which object has a key, what is the key, which other objects can link to this object and through which element or attribute. You can prevent an object to point to an arbitrary object that has an identifier attribute, as it is the case with the ID/IDREF method.
- The scope of key uniqueness is precisely defined among one or several objects within a particular instance of an XML element. It is not more necessary to ensure uniqueness of id attributes across the whole XML document.
- The elements or attributes used as keys or key references can be of any data type, not only ID or IDRef (implying the NMTOKEN format). This allows any element or attribute to be used for linking.

The following principles are taken into account for the implementation of schema identity constraints:

1. Identifiers and references used in schema identity constraints will be attributes. This has the advantage that the data element content of the XML complex types derived from ABIEs is kept unchanged
2. For maximum element and type reuse and to stay away from forward compatibility problems, attributes used as identifiers or references will be optional. This means that no key (xsd:key) constraints should be defined on identifiers, which would make the identifiers mandatory in the context of a message; only uniqueness (xsd:unique) constraints must be used.
3. 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.

[R 8E89]	Schema identity constraints MUST be used to implement references between elements when they represent ABIE's that are linked by an association, whose **`AggregationKind`** property is 'shared'.	1
[R 8103]	The uniqueness (**`xsd:unique`**) constraint MUST be used rather than the key (**`xsd:key`**) constraint to define the keys and enforce that their values are unique within their scope of application.	1

I.2.2 Attribute and Element Declarations

I.2.2.1 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 (**`xsd:key`**/**`xsd:keyref`**).
- To serve as identifiers for dynamic referencing.

[R 8EE7]	Identifiers used in schema identify constraints or for dynamic referencing MUST be declared as attributes.	1
[R 991C]	User defined attributes MUST only be used for Supplementary Components or to serve as identifiers in identity constraints. Modification to Rule [R AFEE].	1

I.2.2.2 Elements

[R A577]	Empty elements MUST NOT be used, except when their definition includes an identifier attribute that serves to reference another element via schema identity constraints. Modification to Rule [R B8B6].	1

I.3 XML Schema Files

I.3.1 Root XML Schema Files

I.3.1.1 Declaration of the Referencing Constraints

Referencing between ABIEs occurs within the limits defined by a particular 'scope' element in the XML document tree.

The scope element is the container of all the elements that can be involved in the identity constraints. The schema language requires that the identity constraints be contained in the schema declaration of the scope element.

Most often the scope element will be the message root element, but it can also be another element lower in the hierarchy.

The identifier attribute of each ABIE that is part of a logical aggregation implemented by XML referencing will be subject to a uniqueness (**`xsd:unique)`** constraint defined in the scope element. The name of the **`xsd:unique`** constraint must be unique in the schema.

The uniqueness (**`xsd:unique`**) constraints define the keys and enforce that a value is unique within the scope element.

The key reference (**`xsd:keyRef`**) constraints define the key references and enforce that a value corresponds to a value represented by a uniqueness (xsd:unique) constraint.

[R BA43]	Each ABIE element that is a scope element of a set of XML Schema identity constraints MUST contain one or more xsd:unique constraint declarations.	1
[R 88DB]	Each ABIE that is the target of a reference under a scope element MUST be the object of a **`xsd:unique`** constraint declaration via a **`xsd:selector/@xpath`** component.	1
[R B40C]	The name of an **`xsd:unique`** constraint MUST be constructed as follows: "**`<Scope element><Referenced Element>Key`**" Where: • Scope element – is the name of the scope element. • Referenced Element – is the element name being referenced within the scope element.	1

This declaration will guarantee uniqueness of the identifier attribute values across all referenced elements of the same name, in the given scope.

[Note:]

The value of **`xsd:selector/@xpath`** identifies instances of one element in one namespace (by default the namespace of the XML Schema File in which the **`xsd:selector`** is declared.).

In Example I-3 the declaration under the message root element will guarantee uniqueness of the **`@key`** attribute values across all **`bie:Party`** elements, in the scope of the **`rsm:ClaimNotify`** message.

Example I-3: Unique Declaration

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

For each referenced ABIE used in a given scope, corresponding key reference (**`xsd:keyRef`**) declarations must be made. Naming conventions used for key reference attributes, as exposed in I.3.2.2, are such that only one key reference (**`xsd:keyRef`**) declaration is needed for all the elements where the key 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 scope element.	1
[R 9BE8]	The **`xsd:keyref/xsd:selector/@xpath`** component must be such that it selects all the elements where the key reference attribute may occur.	1
[R 858D]	The name of an **`xsd:keyref`** constraint MUST be constructed as follows: “**`<Scope Element ><Referenced Element>Reference`**” Where: • Scope Element – is the name of the scope element. • Referenced Element – is the element name being referenced within the scope element.	1

In Example I-4 the declaration under the message root element will enforce referencing between all the elements that have the `@PartyReference` attribute and instances of `bie:Party`, in the scope of the `rsm:ClaimNotify` message.

Example I-4: 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

I.3.2 Business Information Entities XML Schema Files

I.3.2.1 Type Definitions

Every aggregate business information entity (ABIE) **`xsd:complexType`** definition will include an optional identifier attribute that may be used for both dynamic and structural referencing. It will be defined as a local attribute 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, locally defined, “key” attribute that MAY be used as the complex element identifier in the XML document where it appears.	1
[R 92C0]	“**`key`**” MUST be a reserved attribute name.	1
[R 8A37]	Every “**`key`**” local attribute declaration MUST be of the type **`xsd:token.`**	1

I.3.2.2 Element Declarations and References

I.3.2.2.1 ASBIE Elements

For each ASBIE who’s **`ccts:AggregationKind`** value=**`Shared`**, 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.

See section 5.4 Reusability Schema and I.1.1 Message Assembly Considerations earlier this specification.

[R B78E]	Every ASBIE whose **`ccts:AggregationKind`** value=**`Shared`**, 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 B173]	For each equivalent referencing element an **`xsd:complexType`** MUST be declared. Its structure will be an empty element with a local attribute.	1
[R AEDD]	The equivalent referencing element MUST have a name composed of the ASBIE property term and property qualifier term(s)) and the object term and qualifier term(s) of the associated ABIE.	1
[R B3E5]	When there is no ASBIE property term the generic property term "Referred" followed by the name of the associated ABIE MUST be used as a naming convention to distinguish this element from the ABIE element.	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 declared using the **`xsd:complexType`** that relates to the ABIE being referenced.	1

Example I-5 shows the schema definition of an ASBIE specified as a referencing element.

Example I-5: 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"/>
```

Appendix J. Naming and Design Rules List

[R B998]	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:	1

Rule Categorization	
ID	Description
1	Rules which must not be violated by individual organizations else conformance and interoperability is lost – such as named types.
2	Rules which may be modified by individual organizations while still conformant to the NDR structure – such as namespace string contents and namespace tokens.
3	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. (Changes to the XML Schema Architecture.)
4	Rules that if violated lose conformance with the UN/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 UN/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.

[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 Part 2: Datatypes Second Edition.	1
[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 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 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: **URN:** `urn:<organization>:<org hierarchy>[:<org hierarchy level>]*:<schematype>:<context category>:<major>:<status>` **URL:** `http://<organization>/<org hierarchy>[/<org hierarchy level>]*/<schematype>/context category/<major>/<status>`	3

	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.	
[R 8CED]	UN/CEFACT namespaces MUST be defined as Uniform Resource Names.	3
[R B56B]	Published namespace content MUST only be changed by the publishing organization of the namespace or its successor.	1
[R 92B8]	The XML Schema File name for files other than code lists and identifier schemes 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>`** that is expressed in the XML Schema File by using the value of **`<BusinessInformationPayload>`** followed by the words '**`XML Schema File`**' as the name and placing the name in the Header documentation section of the file.	1
[R 9961]	A Root XML Schema File MUST NOT replicate reusable constructs available in XML Schema Files that can be referenced	1

	through **`xsd:include`**.	
[R 8238]	A BIE XML Schema File MUST be created within each namespace that is defined for the primary context category value.	1
[R 8252]	The BIE XML Schema Files MUST be named 'Business Information Entity XML Schema File' by placing the name within the Header documentation section of the file.	1
[R A2F0]	An unqualified BDT XML Schema File MUST be created in the documentation common namespace to represent the set of unrestricted BDTs.	1
[R AA56]	A BDT XML Schema File MUST be created within each namespace that is defined for the primary context category value.	1
[R 847C]	The BDT XML Schema Files MUST be named 'Business Data Type XML Schema File' by placing the name within the header documentation section of the file.	1
[R 8A68]	A Code List XML Schema File MUST be created to convey code list enumerations for each code list being used.	1
[R B0AD]	The name of each Code List XML Schema File as defined in the comment within the XML Schema File MUST be of the form: **`<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>" - Code List XML Schema File"`** Where: • Code List Agency Identifier – Identifies the agency that maintains the code list. • Code List Agency Name – Agency that maintains the code list. • Code List Identification Identifier – Identifies a list of the respective corresponding codes. • Code List Name – The name of the code list as assigned by the agency that maintains the code list.	1
[R 942D]	Each CCL XML Schema File MUST contain enumeration values for both the actual codes and the code values.	1
[R A8A6]	Each BCL XML Schema File MUST contain enumeration values for both the actual codes and the code values, through one of the following: • The restriction of an imported CCL. • The extension of a CCL where the codes and values of the CCL are included and the new extensions are added.	1

	• The creation of a new Code List that is used within the context category value namespace.	
[R AB90]	An Identifier Scheme XML Schema File MUST be created to convey identifier scheme metadata for each scheme being used.	1
[R A154]	The name of each Identifier Scheme XML Schema File as defined in the comment within the XML Schema File MUST be of the form: **`<Identifier Scheme Agency Identifier|Identifier Scheme Agency Name><Identifier Scheme Identification Identifier|Identifier Scheme Name>”  Identifier Scheme XML Schema File”`** Where: • Identifier Scheme Agency Identifier – Identifies the agency that maintains the identifier scheme. • Identifier Scheme Agency Name – Agency that maintains the identifier scheme. • Identifier Scheme Identification Identifier – Identifies the scheme. • Identifier Scheme Name – The name of the identifier scheme as assigned by the agency that maintains the identifier scheme.	1
[R BD2F]	A Business Identifier Scheme XML Schema File MUST be created for each Business Scheme used by a BDT.	1
[R AFEB]	Each Business Identifier Scheme XML Schema File MUST contain metadata that describes the scheme or points to the scheme.	1
[R B564]	Imported XML Schema Files MUST be fully conformant to category 1, 2, 3, 4 and 7 rules as defined in rule [R 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 within an XML Schema File MUST contain a resolvable relative path URL.	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: **`<xsd:schema ... version=” <major>”p”<minor>[”p”<revision>]”>`**	2

	Where: • <major> - sequential number of the major version. • <minor> - sequential number of the minor version • <revision> - optional sequential number of the revision.	
[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 88E2]	Every UN/CEFACT XML Schema File MUST use UTF-8 encoding.	1
[R ABD2]	Every XML Schema File MUST contain a comment that identifies its name immediately following the XML declaration using the format defined in Appendix B-2.	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 header comment within the XML Schema File.	1

[R 9443]	The CCTS Metadata XML Schema File MUST reside in its own namespace and be defined in accordance with rule [R 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 B20F]	**`xsd:redefine`** MUST NOT be used.	4 6
[R 926D]	**`xsd:substitutionGroup`** MUST NOT be used.	4 6
[R 8A83]	**`xsd:ID/xsd:IDREF`** MUST NOT be used.	1
[R B221]	Supplementary Components MUST be declared as Attributes.	1
[R AFEE]	User defined attributes MUST only be used for Supplementary Components.	3
[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 identifiers MUST be based on the **`xsd:simpleType`** of the appropriate identifier scheme.	1
[R B8B6]	Empty elements MUST NOT be used.	3
[R 8337]	The **`xsd:nillable`** attribute MUST NOT be used.	1
[R 8608]	Anonymous types MUST NOT be used.	1
[R A4CE]	An **`xsd:complexType`** MUST be defined for each CCTS ABIE.	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 BDT XML Schema File.	4 6
[R 9D6E]	**`xsd:extension`** MUST only be used for declaring **`xsd:attribute`**s to accommodate relevant supplementary components.	4 6
[R 9947]	**`xsd:restriction`** MUST only be used in BDT XML Schema Files.	1
[R 8AF7]	When **`xsd:restriction`** is applied to a data type the resulting type MUST be uniquely named.	1
[R 847A]	Each defined or declared construct MUST use the **`xsd:annotation`** element for required CCTS documentation and application information to communicate context.	1
[R A9EB]	Each defined or declared construct MUST use an **`xsd:annotation`** and **`xsd:documentation`** element for required CCTS documentation.	3
[R 9B07]	Each of the resulting XML Schema Components (**`xsd:element`**, **`xsd:complexType`** and **`xsd:simpleType`**) MUST have an **`xsd:annotation xsd:appInfo`** declared that includes zero or more **`ccts:UsageRule`** elements and one or more **`ccts:BusinessContext`** elements.	1
[R 88DE]	Usage rules MUST be expressed within an **`xsd:appInfo ccts:UsageRule`** element.	1
[R B851]	The structure of the **`ccts:UsageRule`** element MUST be: • **`ccts:UniqueID`** [1..1] – A unique identifier for the UsageRule. • **`ccts:Constraint`** [1..1] – The actual constraint expression. • **`ccts:ConstraintType`** [1..1] – The type of constraint E.g. unstructured, OCL. • **`ccts:ConditionType`** [1..1] – The type of condition. Allowed values are **`pre-condition`**, **`post-condition`**, and **`invariant`**.	1
[R A1CF]	A ccts:ConstraintType code list XML Schema File will be created.	1
[R A538]	Each defined or declared XML Schema artifact MUST use an **`xsd:annotation`** and **`xsd:appInfo`** element to	1

	communicate the context of the artifact.	
[R B96F]	Each Root, BIE, BDT and BCL XML Schema File MUST be assigned to a unique namespace that represents the primary context category value of its contents.	1
[R B698]	The Root XML Schema File MUST include the BIE and BDT XML Schema Files that reside in its 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 using the XML Schema Component **`xsd:element`**.	1
[R A466]	The name of the root element MUST be the same as the name of the business information payload data dictionary name, with separators and spaces removed.	1
[R 8062]	The root element declaration MUST be defined using an **`xsd:complexType`** that represents the message content contained within the business information payload.	1
[R 8837]	Each Root XML Schema File MUST define a single **`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 8010]	The Root XML Schema File root element declaration MUST have a structured set of annotations documentation (**`xsd:annotation xsd:documentation`**) present in that includes: • UniqueID (mandatory): The identifier that uniquely identifies the business information payload, the root element. • VersionID (mandatory): The unique identifier that identifies the version of the business information payload, the root element. • ObjectClassQualifierName (zero or more): Is a word or words which help define and differeniate an ABIE from its associated CC and other BIEs. It enhances the sematic meaning of the DEN to reflect a restriction of the concept, conceptual domain, content model or data value. • ObjectClassTermName (mandatory): Is a semantically meaningful name of the Object class. It is the basis for the DEN. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the business information payload. • Definition (mandatory): The semantic meaning of the root	1

	element. • BusinessTermName (optional, repeating): A synonym term under which the payload object is known by in industry.	
[R 8FE2]	The BIE XML Schema File MUST contain an **`xsd:include`** statement for the BDT XML Schema File that resides in the same namespace.	1
[R AF95]	For every object class (ABIE) identified in a primary context category, a named **`xsd:complexType`** MUST be defined in its corresponding BIE XML Schema File.	1
[R 9D83]	The name of the ABIE **`xsd:complexType`** MUST be the **`ccts:DictionaryEntryName`** with the spaces and separators removed, with approved abbreviations and acronyms applied and with the '**`Details`**' suffix replaced with '**`Type`**'.	1
[R 90F9]	The cardinality and sequencing of the elements within an ABIE **`xsd:complexType`** MUST be as defined by the corresponding ABIE values in the syntax neutral model.	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 A21A]	Every BBIE within the containing ABIE MUST have a named **`xsd:simpleType`** (If the BBIE BDT includes only the content component) or **`xsd:complexType`** (If the BBIE BDT includes one or more supplementary components).	1
[R 8B85]	Every BBIE type MUST be named the property term and qualifiers and the representation term of the basic business information entity (BBIE) it represents with the word '**`Type`**' appended.	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 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 BBIE.	1
[R 96D9]	For 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]	For 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 BCD6]	Every BBIE element declaration MUST be of the BusinessDataType that represents the source basic business information entity (BBIE) data type.	1
[R 9025]	For every ASBIE whose **`ccts:AggregationKind`** value = **`composite`**, a local element for the associated ABIE MUST be declared in the associating ABIE **`xsd:complexType`** content model.	1
[R 9241]	For every ASBIE whose **`ccts:AggregationKind`** value = **`shared`**, a global element MUST be declared.	1
[R A08A]	Each ASBIE 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]	Each ASBIE element declaration MUST use the **`xsd:complexType`** that represents its associated ABIE.	1
[R ACB9]	For every ABIE **`xsd:complexType`** definition a structured set of annotations MUST be present in the following pattern: • UniqueID (mandatory): The unique identifier that identifies an ABIE instance in a unique and unambiguous way. • VersionID (mandatory): An unique identifier that identifies the version of an ABIE. • ObjectClassQualifierName (optional, repeating): Is a word or ordered words which help define and differeniate the	1

	associated ABIE from its CC. • ObjectClassTermName (mandatory): Is a semantically meaningful name of the object class of the ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the ABIE. • Definition (mandatory): The semantic meaning of the ABIE. • BusinessTermName (optional, repeating): A synonym term in which the ABIE is commonly known.	
[R B0BA]	For every ABIE **`xsd:complexType`** definition a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements MUST be present that fully declare its context.	1
[R BCE9]	For every ABIE usage rule, the ABIE **`xsd:complexType`** definition MUST contain a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements in the following pattern: • **`ccts:UniqueID`** • **`ccts:Constraint`** • **`ccts:ConstraintType`** • **`ccts:ConditionType`**.	1
[R 88B6]	For every ABIE **`xsd:element`** declaration definition, a structured set of annotations MUST be present in the following pattern: • UniqueID (mandatory): The unique identifier that identifies an ABIE instance in a unique and unambiguous way. • VersionID (mandatory): An unique identifier that identifies the version of an ABIE. • ObjectClassQualifierName (optional, repeating): Is a word or ordered words which help define and differeniate the associated ABIE from its CC. • ObjectClassTermName (mandatory): Is a semantically meaningful name of the object class of the ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the ABIE. • Definition (mandatory): The semantic meaning of the ABIE. • BusinessTermName (optional, repeating): A synonym term in which the ABIE is commonly known.	1
[R B8BE]	For every BBIE **`xsd:element`** declaration a structured set of **`xsd:annotation`** **`xsd:documentation`** elements MUST be present in the following pattern:	1

	• Cardinality (mandatory): Indicates the cardinality of the BBIE within the containing ABIE. • SequencingKey (mandatory): Indicates the sequence of the BBIE within the containing ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the BBIE. • Definition (mandatory): The semantic meaning of the associated BBIE. • BusinessTermName (optional, repeating): A synonym term in which the BBIE is commonly known. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the BBIE. • PropertyQualifierName (optional repeating): Is a word or words which help define and differentiate the BBIE. • RepresentationTermName (mandatory): An element of the component name that describes the form in which the BBIE is represented.	
[R 95EB]	For every BBIE **`xsd:element`** declaration a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements MUST be present that fully declare its context.	1
[R 8BF6]	For every BBIE usage rule, the BBIE xsd:element declaration MUST contain a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements in the following pattern: • **`ccts:UniqueID`** • **`ccts:Constraint`** • **`ccts:ConstraintType`** • **`ccts:ConditionType`**.	1
[R 8D3E]	Every ASBIE global element declaration MUST have a structured set of **`xsd:annotation xsd:documentation`** elements in the following pattern: • AssociationKind (mandatory): Indicates the UML AssociationKind value of **`shared`** or **`composite`** of the associated ABIE. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the ASBIE. • PropertyQualifierName (optional repeating): Is a word or words which help define and differentiate the ASBIE. • AssociatedObjectClassName (Mandatory): The name of the	1

	associated object class. • AssociatedObjectClassQualifierName (optional, repeating): a name or names that qualify the associated object class.	
[R 926A]	Every ASBIE **`xsd:element`** declaration or **`xsd:ref`** occurence MUST have a structured set of **`xsd:annotation`** **`xsd:documentation`** elements present in the following pattern: • Cardinality (mandatory): Indicates the cardinality of the ASBIE within the containing ABIE. • SequencingKey (mandatory): Indicates the sequence of the ASBIE within the containing ABIE. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the ASBIE. • Definition (mandatory): The semantic meaning of the ASBIE. • BusinessTermName (optional, repeating): A synonym term in which the ASBIE is commonly known. • AssociationKind (mandatory): Indicates the UML AssociationKind value of **`shared`** or **`composite`** of the associated ABIE. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the ASBIE. • PropertyQualifierName (optional repeating): Is a word or words which help define and differentiate the ASBIE. • AssociatedObjectClassName (Mandatory): The name of the associated object class. • AssociatedObjectClassQualifierName (optional, repeating): a name or names that qualify the associated object class.	1
[R 9D87]	Every ASBIE **`xsd:element`** declaration or ASBIE **`xsd:ref`** to an ABIE global element declaration MUST contain a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements that fully declare its context.	1
[R A76D]	Every ASBIE usage rule **`xsd:element`** declaration or ASBIE **`xsd:ref`** to an ABIE global element declaration MUST contain a structured set of **`xsd:annotation`** **`xsd:appInfo`** elements in the following pattern: • **`ccts:UniqueID`** • **`ccts:Constraint`** • **`ccts:ConstraintType`**	1

	• **ccts:ConditionType**.	
[R 8E0D]	The BDT XML Schema File MUST include (**xsd:include**) the BCL XML Schema Files and BIS XML Schema Files that are defined in the same namespace.	1
[R B4C0]	The BDT XML Schema File MUST import (**xsd:import**) the CCL XML Schema Files and CIS XML Schema Files that are used by a BDT contained within the file.	1
[R AE00]	Each CCTS BDT artifact within the UN/CEFACT Data Type Catalogue used by the Root XML Schema Files and the BIE XML Schema File within a given namespace MUST be defined as an **xsd:simpleType** or **xsd:complexType** in the BDT XML Schema File with the given namespace.	1
[R 9908]	For every BDT whose content component BVD is defined by a primitive whose facets map directly to the facets of an XSD built-in data type, the BDT MUST be defined as a named **xsd:simpleType**.	1
[R B91F]	Every BDT whose content component BVD is defined by a primitive whose facets map directly to the facets of an **xsd:simpleType** MUST contain one **xsd:restriction** element.	1
[R 9910]	The **xsd:restriction** element used in a BDT content component BVD defined by a primitive MUST include an **xsd:base** attribute that defines the specific XSD built-in data type required for the content component.	1
[R A7B8]	The name of a BDT that uses a primitive to define its content component BVD MUST be the BDT **ccts:DataTypeQualifier(s)** if any, plus the **ccts:DataTypeTerm**, plus the primitive type name, followed by the word '**Type**' with the separators removed and approved abbreviations and acronyms applied.	1
[R AA60]	A BDT whose content component BVD is defined as an **xsd:simpleType** whose base is a single code list MUST contain an **xsd:restriction** element with the **xsd:base** attribute set to the code lists defined xsd:simpleType.	1
[R 8DB1]	The name of A BDT that uses a single code list to define its content component BVD MUST be its **ccts:DataTypeQualifier(s)** if any, plus the **ccts:DataTypeTerm**, plus the code list suffix, followed by the	1

	word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied. The code list suffix MUST be the following: (Any repeated words are eliminated.) • **`<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>`** Where. • Code List Agency Identifier – is the identifier for the agency that code list is from. • Code List Agency Name – is the name of the agency that maintains the code list. • Code List Identification Identifier – is the identifier for the given code list. • Code List Name – is the name for the code list.	
[R AAD1]	A BDT whose content component BVD is defined by a choice of two or more code lists MUST be defined as an **`xsd:simpleType`** that contains an **`xsd:union`** element whose **`xsd:memberType`** attribute includes the **`xsd:simpleType`** definitions of the code lists to be included.	1
[R 973C]	The name of a BDT that uses multiple code lists MUST be it's **`ccts:DataTypeQualifier(s)`** if any, plus the **`ccts:DataTypeTerm`**, plus the code list suffix, followed by the word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied. The suffix MUST be the following: (Any repeated words are eliminated) • **`<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>`** Where: • Code List Agency Identifier – is the identifier for the agency that code list is from. • Code List Agency Name – is the name of the agency that maintains the code list. • Code List Identification Identifier – is the identifier for the given code list. • Code List Name – is the name for the code list.	1

[R A861]	If a BDT content component BVD is defined as an **`xsd:simpleType`** whose base is an identifier scheme, it MUST contain an **`xsd:restriction`** element with the **`xsd:base`** attribute set to the identifier scheme defined **`xsd:simpleType`**.	1
[R 8F96]	The name of A BDT that uses an identifier scheme to define its content component BVD MUST be its **`ccts:DataTypeQualifier(s)`** if any, plus the **`ccts:DataTypeTerm`**, plus the identifier scheme suffix, followed by the word '**`Type`**' with the separators removed and approved abbreviations and acronyms applied.. The code list suffix MUST be the following: (Any repeated words are eliminated.) • **`<Identifier Scheme Agency Identifier|Identifier Scheme Agency Name><Identifier Scheme Identification Identifier|Identifier Scheme Name>`** Where. • Identifier Scheme Agency Identifier – is the identifier for the agency that code list is from. • Identifier Scheme Agency Name – is the name for the Agency that owns the identifier scheme. • Identifier Scheme Identification Identifier – is the identifier for the given identifier scheme. • Identifier Scheme Name – is the name for the identifier scheme.	1
[R AB05]	Every BDT that includes one or more Supplementary Components MUST be defined as an **`xsd:complexType`**	1
[R AAA5]	Every BDT **`xsd:complexType`** definition MUST have an **`xsd:simpleContent`** expression whose **`xsd:extension base`** attribute is set to the primitive type or scheme or list that defines its Content Component Business Value Domain.	1
[R 890A]	Every BDT **`xsd:complexType`** definition MUST include an **`xsd:attribute`** declaration for each Supplementary Component.	1
[R ABC1]	The name of the Supplementary Component xsd:attribute must be the DEN of the Supplementary Component with periods, spaces, and other separators removed.	1
[R 90FB]	The name of a BDT that includes one or more Supplementary Components MUST be:	1

	• The BDT **`ccts:DataTypeQualifier(s)`** if any, plus • The **`ccts:DataTypeTerm`**, plus • The suffix of the Content Component Business Value Domain where: o The suffix is the primitive type name, the code list token, the series of code list tokens, or the identifier scheme token. plus • The **`ccts:DictionaryEntryName`** for each Supplementary Component present following the order defined in the Data Type Catalogue, plus • The suffix that represents the Supplementary Component BVD where the suffix is the primitive type name, the code list token, the series of code list tokens, or the identifier scheme token, plus • The word '**Type**'. • With all separators removed and approved abbreviations and acronyms applied.	
[R 80FD]	Every restricted BDT 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 BDT XML Schema Component **`xsd:type`** definition requiring a non-standard variation from its base type MUST be derived from a custom type.	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]	In the BDT XML Schema File, local **`xsd:attribute`** declarations MUST only represent CCTS Supplementary Components for the BDT for which they are declared.	1
[R BFE5]	Every BDT definition MUST contain a structured set of annotation documentation in the following sequence and pattern: • UniqueID (mandatory): The unique identifier that identifies the BDT in a unique and unambiguous way. • VersionID (mandatory): An unique identifier that identifies	1

	the version of the BDT. • DictionaryEntryName (mandatory): The Data Dictionary Entry Name (DEN) of the BDT. • Definition (mandatory): The semantic meaning of the BDT. • BusinessTermName (optional, repeating): A synonym term in which the BDT is commonly known. • PropertyTermName (mandatory): Represents a distinguishing characteristic of the BDT and shall occur naturally in the definition. • DataTypeName (mandatory): The name of the DataType. The possible values for the DataType are defined in the Data Type Catalogue. • DataTypeQualifierName (mandatory): Is a word or words which help define and differentiate a Data Type. It further enhances the semantic meaning of the DataType. • DefaultIndicator (mandatory): Indicates that the specific Code List Value is the default for the Code List. • DefaultValue (optional): Is the default value. • DefaultValueSource (optional): Indicates the source for the default value. • SchemeOrListID (optional): The unique identifier assigned to the scheme or list that uniquely identifies it. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the Scheme or Code List being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the Scheme or Code List being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the enumertations specified by the Scheme or Code List. • SchemeOrListName (optional): Name of the Scheme or Code List. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the Scheme or Code List is commonly known and used in business. (BusinessTerm)	
[R 9C95]	Every supplementary component `xsd:attribute` declaration MUST contain a structured set of annotation documentation MUST in the following pattern: • Cardinality (mandatory): Indicates the cardinality of the SC within the containing BDT.	1

	• PropertyTermName (mandatory): Represents a distinguishing characteristic of the SC and shall occur naturally in the definition. • RepresentationTermName (mandatory): An element of the component name that describes the form in which the SC is represented. • PrimitiveTypeName (mandatory): The name of the SC PrimitiveType. • DataTypeName (mandatory): The name of the DataType. The possible values for the DataType are defined in the Data Type Catalogue. • DataTypeQualifierName (mandatory): A word or words which help define and differentiate a Data Type. It further enhances the semantic meaning of the DataType. • DefaultIndicator (mandatory): Indicates that the specific Code List Value is the default for the Code List or identifier scheme. • DefaultValue (optional): Is the default value. • DefaultValueSource (optional): Indicates the source for the default value. • SchemeOrListID (optional): The unique identifier assigned to the scheme or list that uniquely identifies it. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the identifier scheme or code list being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the identifier scheme or code list being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the enumertations specified by the identifier scheme or code list. • SchemeOrListName (optional): Name of the identifier scheme or code list. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the identifier scheme or code list is commonly known and used in business. (BusinessTerm)	
[R 9E40]	Each code list used by a BDT or BBIE MUST be defined in its own XML Schema File.	2

[R 849E]	Code List XML Schema File names MUST be of the form: **`<Agency Identifier | Agency Name>_<List Identification Identifier | List Name>_<Version Identifier>.xsd`** All periods, spaces, or other separators are removed except for the "." before xsd and the "_" between the names. Where: • 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. • Agency Name – the name of the agency that maintains the list. • List Identification Identifier – identifies a list of the respective corresponding codes or ids. • List Name – the name of a list of codes. • Version Identifier – identifies the version.	2
[R 8D1D]	Each Code List XML Schema File MUST declare a single global element.	3
[R BE84]	The Code List XML Schema File global element MUST be of the **`xsd:simpleType`** that is defined in the Code List XML Schema File.	3
[R A8EF]	Each Code List XML Schema File MUST define one, and only one, named **`xsd:simpleType`** for the content component.	1
[R 92DA]	The Code List XML Schema File **`xsd:simpleType`** name MUST be the name of the code list root element with the word '**`ContentType`**' appended.	1
[R 962C]	Each code in a Code List XML Schema File MUST be expressed as **`xsd:enumeration`**, where the **`xsd:value`** for the enumeration is the actual code value.	1
[R A142]	Every Code List MUST contain a structured set of annotation documentation in the following sequence and pattern: • SchemeOrListID (mandatory): The unique identifier assigned to the code list. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the code list being referenced. • SchemeOrListAgencyName (optional): The name of the	1

	Agency that owns or is responsible for the code list being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the enumerations specified by the code list. • SchemeOrListName (optional): Name of the code list. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the code list is commonly known and used in business. (BusinessTerm)	
[R A814]	Each code list **`xsd:enumeration`** MUST contain a structured set of annotations in the following sequence and pattern: • Name (mandatory): The name of the code. • Description (optional): Descriptive information concerning the code.	1
[R 992A]	Code list XML Schema File namespaces MUST use the following pattern: **URN:** **`urn:<organization>:<org hierarchy>*[:<org hierarchy level n>]:codelist:common:<major>:<status>:<name>`** **URL:** **`http://<organization>/<org hierarchy>*[/<org hierarchy level n>]/codelist/common/<major>/<status>/<name>`** Where: • organization – 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. • codelist – A fixed value token for common codelists. • common – A fixed value token for common codelists. • major – The Major version number of the codelist. • status – The status of the schema as: draft|standard • name – The name of the XML Schema File (using upper camel case) with periods, spaces, or other separators and the words 'schema module' removed. o Code list names are further defined as: <Code List	1

	Agency Identifier|Code List Agency Name> ><divider><Code List Identification Identifier|Code List Name> Where: ▪ Code List Agency Identifier – is the identifier for the agency that code list is from. ▪ Code List Agency Name – is the name of the agency that maintains the code list. ▪ Divider – the divider character for URN is ':' the divider character for URL is '/'. ▪ Code List Identification Identifer – is the identifier for the given code list. ▪ Code List Name – is the name for the code list.	
[R 9FD1]	Each UN/CEFACT maintained CCL XML Schema File MUST be represented by a unique token constructed as follows: **`clm<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>`** Such that any repeated words are eliminated. Where: • Code List Agency Identifier – is the identifier for the agency that code list is from. • Code List Agency Name – is the name of the agency that maintains the code list. • Code List Identification Identifier – is the identifier for the given code list. • Code List Name – is the name for the code list.	2
[R 86C8]	CCL XML Schema Files MUST NOT import or include any other XML Schema Files.	1
[R B40B]	Each CCL XML Schema File **`xsd:simpleType`** MUST use an **`xsd:restriction`** element whose base attribute is **`xsd:token`**.	1
[R 8F2D]	BCL XML Schema file MUST be used to • Extend existing CCL or • Define a codelist where one does not exist or • Restrict the value of a CCL for a context category	1

[R 87A9]	BCL XML Schema Files MUST import only CCL XML Schema Files it uses directly.	1
[R 882D]	In each BCL XML Schema File the **`xsd:restriction`** element base attribute value MUST be set to **`xsd:token`**.or the '**`ContentType`**' from the CCL that is being used.	1
[R A1EE]	Each identifier scheme used by a BDT or BBIE MUST be defined in its own XML Schema file.	2
[R A50B]	Identifier Scheme XML Schema File names MUST be of the form: **`<Agency Identifier | Agency Name>_<Scheme Identification Identifier | Scheme Name>_<Version Identifier>.xsd`** All periods, spaces, or other separators are removed except for the "." before xsd and the "_" between the names. Where: • Agency Identifier – identifies the agency that manages the identifier scheme. The default agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used. • Agency Name – the name of the agency that maintains the scheme. • Scheme Identification Identifier – identifies the identifier scheme. • Scheme Name – the name of the identifier scheme. • Version Identifier – identifies the version of the scheme.	2
[R BFEB]	Each Identifier Scheme XML Schema File MUST declare a single global element.	3
[R B236]	The Identifier Scheme XML Schema File root element MUST be of the **`xsd:simpleType`** that is defined in the Identifier Scheme XML Schema File.	3
[R 9451]	Each Identifier Scheme XML Schema File MUST define one, and only one, named **`xsd:simpleType`** for the content component.	1
[R 8CD3]	The Identifier Scheme XML Schema File **`xsd:simpleType`** name MUST be the name of the identifier scheme root element with the word '**`ContentType`**' appended.	1

[R B30A]	Every Identifier Scheme MUST contain a structured set of annotation documentation in the following sequence and pattern: • SchemeOrListID (mandatory): The unique identifier assigned to the Identifier Scheme. • SchemeOrListAgencyID (optional): The unique identifier assigned to the Agency that owns or is responsible for the identifier scheme being referenced. • SchemeOrListAgencyName (optional): The name of the Agency that owns or is responsible for the identifier scheme being referenced. • SchemeOrListModificationAllowedIndicator (optional): Indicates whether the values being validated can be outside the pattern specified by the scheme. • SchemeOrListName (optional): Name of the identifier scheme. • SchemeOrListBusinessTermName (optional, repeating): A synonym term under which the identifier scheme is commonly known and used in business. (BusinessTerm)	1
[R 9CCF]	Identifier scheme XML Schema File namespaces MUST use the following pattern: **URN:** `urn:<organization>:<org hierarchy>*[:<org hierarchy level n>]:identifierscheme:common:<major>:<status>:<name>` **URL:** `http://<organization>/<org hierarchy>*[/<org hierarchy level n>]/identifierscheme/common/<major>/<status>/<name>` Where: • organization – 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. • identifierscheme – A fixed value token for common identifier schemes. • common – A fixed value token for common identifier schemes. • major – The Major version number of the identifier scheme.	1

	• status – The status of the schema as: draft|standard • name – The name of the XML Schema File (using upper camel case) with periods, spaces, or other separators and the words ‘schema module’ removed. o Identifier scheme names are further defined as: <Identifier Scheme Agency Identifier|Identifier Scheme Agency Name> ><divider><Identifier Scheme Identification Identifier|Identifier Scheme Name> Where: ▪ Identifier Scheme Agency Identifier – is the identifier for the agency that identifier scheme is from. ▪ Identifier Scheme Agency Name – is the name of the agency that maintains the identifier scheme. ▪ Divider – the divider character for URN is ‘:’ the divider character for URL is ‘/’. ▪ Identifier Scheme Identification Identifer – is the identifier for the given identifier scheme. ▪ Identifier Scheme Name – is the name for the identifier scheme.	
[R B2BC]	Each UN/CEFACT maintained CIS XML Schema File MUST be represented by a unique token constructed as follows: **`clm<Identifier Scheme Agency Identifier|Identifier Scheme Agency Name><Identifier Scheme Identification Identifier|Identifier Scheme Name>`** Such that any repeated words are eliminated. Where: • Identifier Scheme Agency Identifier – is the identifier for the agency that the identifier scheme is from. • Identifier Scheme Agency Name – is the name of the agency that maintains the identifier scheme. • Identifier Scheme Identification Identifier – is the identifier for the given identifier scheme. • Identifier Scheme Name – is the name for the identifier scheme.	2
[R A6C0]	CIS XML Schema Files MUST NOT import or include any other XML Schema Files.	1

[R 9DDA]	Each CIS XML Schema File xsd:simpleType MUST use an **`xsd:restriction`** element whose base attribute value = **`xsd:token`**.	1
[R A1E3]	BIS XML Schema file MUST be used to • Define an identifier scheme where one does not exist or • Redefine an existing CIS	1
[R A4BF]	BIS XML Schema Files MUST NOT use **`xsd:import`** or **`xsd:include`**.	1
[R 96B0]	Each CIS XML Schema File xsd:simpleType MUST use an **`xsd:restriction`** element whose base attribute value is **`xsd:token`**.	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
[R A884]	The attributes for scheme or list supplementary components SHOULD NOT be used within an XML Instance.	1

4387 Naming and Design Rules for the Alternative Business 4388 Message Syntax in Appendix I

[R 8E89]	Schema identity constraints MUST be used to implement references between elements when they represent ABIE's that are linked by an association, whose **`AggregationKind`** property is 'shared'.	1
[R 8103]	The uniqueness (**`xsd:unique`**) constraint MUST be used rather than the key (**`xsd:key`**) constraint to define the keys and enforce that their values are unique within their scope of application.	1
[R 8EE7]	Identifiers used in schema identify constraints or for dynamic	1

	referencing MUST be declared as attributes.	
[R 991C]	User defined attributes MUST only be used for Supplementary Components or to serve as identifiers in identity constraints. Modification to Rule [R AFEE].	1
[R A577]	Empty elements MUST NOT be used, except when their definition includes an identifier attribute that serves to reference another element via schema identity constraints. Modification to Rule [R B8B6].	1
[R BA43]	Each ABIE element that is a scope element of a set of XML Schema identity constraints MUST contain one or more xsd:unique constraint declarations.	1
[R 88DB]	Each ABIE that is the target of a reference under a scope element MUST be the object of a **`xsd:unique`** constraint declaration via a **`xsd:selector/@xpath`** component.	1
[R B40C]	The name of an **`xsd:unique`** constraint MUST be constructed as follows: "**`<Scope element><Referenced Element>Key`**" Where: • Scope element – is the name of the scope element. • Referenced Element – is the element name being referenced within the scope element.	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 scope element.	1
[R 9BE8]	The **`xsd:keyref/xsd:selector/@xpath`** component must be such that it selects all the elements where the key reference attribute may occur.	1
[R 858D]	The name of an **`xsd:keyref`** constraint MUST be constructed as follows: "**`<Scope Element ><Referenced Element>Reference`**" Where: • Scope Element – is the name of the scope element. • Referenced Element – is the element name being referenced within the scope element.	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	1

	(e.g. UUIDs).	
[R 8EA2]	Every aggregate business information entity (ABIE) **`xsd:complexType`** definition MUST contain an optional, locally defined, “key” attribute that MAY be used as the complex element identifier in the XML document where it appears.	1
R 92C0]	“**`key`**” MUST be a reserved attribute name.	1
[R 8A37]	Every “**`key`**” local attribute declaration MUST be of the type **`xsd:token.`**	1
[R B78E]	Every ASBIE whose **`ccts:AggregationKind`** value=**`Shared`**, 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 B173]	For each equivalent referencing element an **`xsd:complexType`** MUST be declared. Its structure will be an empty element with a local attribute.	1
[R AEDD]	The equivalent referencing element MUST have a name composed of the ASBIE property term and property qualifier term(s)) and the object term and qualifier term(s) of the associated ABIE.	1
[R B3E5]	When there is no ASBIE property term the generic property term “Referred” followed by the name of the associated ABIE MUST be used as a naming convention to distinguish this element from the ABIE element.	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 declared using the **`xsd:complexType`** that relates to the ABIE being referenced.	1

4389

Appendix K. 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

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 (CCL) – 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.

Core Component Type (CCT) –

Core Data Type (CDT) – The Core Data Type is the data type that constitutes the value space for the allowed values for a 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.

Scope element – (for identity constraints) – The element whose schema declaration contains the identity constraints.

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

Copyright Statement