



# Universal Business Language (UBL) Naming and Design Rules 2.0

Public Review Draft, 8 September 2006

**Document identifier:**

prd-UBL-NDR-2.0

**Location:**

<http://docs.oasis-open.org/ubl/prd-UBL-NDR-2.0>

**Technical Committee:**

OASIS Universal Business Language Technical Committee

**Chairs:**

Jon Bosak, Sun Microsystems

Tim McGrath

**Editors:**

Mavis Cournane, Cognitran Limited <mavis.Cournane@cognitran.com>

Mike Grimley, US Navy <MJGrimley@acm.org>

**Abstract:**

This specification documents the naming and design rules and guidelines for the construction of XML components for the UBL vocabulary.

**Status:**

This document was last revised or approved by the UBL TC on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at [www.oasis-open.org/committees/ubl](http://www.oasis-open.org/committees/ubl).

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page ([www.oasis-open.org/committees/ubl/ipr.php](http://www.oasis-open.org/committees/ubl/ipr.php)).

The non-normative errata page for this specification is located at [www.oasis-open.org/committees/ubl](http://www.oasis-open.org/committees/ubl).

Copyright © 2006 The Organization for the Advancement of Structured Information Standards [OASIS]

37 **Table of Contents**

38 1 Introduction ..... 5

39 1.1 Audiences ..... 5

40 1.2 Scope ..... 5

41 1.3 Terminology and Notation ..... 6

42 1.4 Guiding Principles ..... 7

43 1.4.1 Adherence to General UBL Guiding Principles ..... 7

44 1.4.2 Design for Extensibility ..... 8

45 1.4.3 Relationship to Tools ..... 8

46 1.4.4 Choice of Schema Language ..... 9

47 2 Relationship to ebXML Core Components ..... 10

48 2.1 Mapping Business Information Entities to XSD ..... 12

49 3 General XML Constructs ..... 15

50 3.1 Overall Schema Structure ..... 15

51 3.1.1 Element declarations within document schemas ..... 16

52 3.2 Naming and Modeling Constraints ..... 17

53 3.2.1 Naming Constraints ..... 17

54 3.2.2 Modeling Constraints ..... 17

55 3.3 Reusability Scheme ..... 18

56 3.4 Extension Scheme ..... 19

57 3.5 Namespace Scheme ..... 20

58 3.5.1 Declaring Namespaces ..... 20

59 3.5.2 Namespace Uniform Resource Identifiers ..... 20

60 3.5.3 Schema Location ..... 21

61 3.5.4 Persistence ..... 21

62 3.6 Versioning Scheme ..... 21

63 3.7 Modularity Strategy ..... 24

64 3.7.1 UBL Modularity Model ..... 24

65 3.7.2 Internal and External Schema Modules ..... 28

66 3.7.3 Internal Schema Modules ..... 28

67 3.7.4 External Schema Modules ..... 28

68 3.8 Annotation and Documentation Requirements ..... 32

69 3.8.1 Schema Annotation ..... 32

70 3.8.2 Embedded documentation ..... 32

71 4 Naming Rules ..... 36

72	4.1	General Naming Rules .....	36
73	4.2	Type Naming Rules .....	38
74	4.2.1	Complex Type Names for CCTS Aggregate Business Information Entities (ABIEs)	
75		38	
76	4.2.2	Complex Type Names for CCTS Basic Business Information Entity (BBIE) Properties	
77		38	
78	4.3	Element Naming Rules.....	40
79	4.3.1	Element Names for CCTS Aggregate Business Information Entities (ABIEs).....	40
80	4.3.2	Element Names for CCTS Basic Business Information Entity (BBIE) Properties ...	40
81	4.3.3	Element Names for CCTS Association Business Information Entities (ASBIEs) ....	41
82	4.4	Attributes in UBL.....	41
83	5	Declarations and Definitions .....	42
84	5.1	Type Definitions.....	42
85	5.1.1	General Type Definitions .....	42
86	5.1.2	Simple Types .....	42
87	5.1.3	Complex Types .....	43
88	5.2	Element Declarations .....	45
89	5.2.1	Elements Bound to Complex Types.....	45
90	5.2.2	Elements Representing ASBIEs .....	46
91	5.2.3	Code List Import.....	46
92	5.2.4	Empty Elements .....	46
93	6	Code Lists .....	47
94	7	Miscellaneous XSD Rules.....	48
95	7.1	xsd:simpleType.....	48
96	7.2	Namespace Declaration .....	48
97	7.3	xsd:substitutionGroup.....	48
98	7.4	xsd:final.....	48
99	7.5	xsd: notation .....	48
100	7.6	xsd:all.....	49
101	7.7	xsd:choice.....	49
102	7.8	xsd:include.....	49
103	7.9	xsd:union .....	49
104	7.10	xsd:appinfo .....	49
105	7.11	xsd:schemaLocation .....	50
106	7.12	xsd:nillable .....	50
107	7.13	xsd:anyAttribute.....	50
108	7.14	Extension and Restriction.....	50

109	8 Instance Documents .....	51
110	Appendix A. UBL NDR 2.0 Checklist.....	52
111	A.1 Attribute Declaration rules .....	52
112	A.2 Code List rules.....	53
113	A.3 ComplexType Definition rules.....	53
114	A.4 Complex Type Naming rules .....	54
115	A.5 Documentation rules.....	55
116	A.6 Element Declaration rules.....	58
117	A.7 Element Naming rules .....	59
118	A.8 General Naming rules.....	60
119	A.9 General Type Definition Rules.....	60
120	A.10 General XML Schema Rules .....	61
121	A.11 Modelling constraint rules.....	63
122	A.12 Naming constraint rules.....	64
123	A.13 Namespace Rules .....	64
124	A.14 Root element declaration rules.....	65
125	A.15 Schema structure modularity rules .....	65
126	A.16 Standards Adherence rules .....	66
127	A.17 Versioning rules .....	66
128	Appendix B. Technical Terminology.....	68
129	Appendix C. References.....	73
130	Appendix D. Notices .....	74

---

131

# 1 Introduction

132 XML is often described as the lingua franca of e-commerce. The implication is that by  
133 standardizing on XML, enterprises will be able to trade with anyone, any time, without the need  
134 for the costly custom integration work that has been necessary in the past. But this vision of XML-  
135 based “plug-and-play” commerce is overly simplistic. Of course XML can be used to create  
136 electronic catalogs, purchase orders, invoices, shipping notices, and the other documents needed  
137 to conduct business. But XML by itself doesn't guarantee that these documents can be  
138 understood by any business other than the one that creates them. XML is only the foundation on  
139 which additional standards can be defined to achieve the goal of true interoperability. The  
140 Universal Business Language (UBL) initiative is the next step in achieving this goal.

141 The task of creating a universal XML business language is a challenging one. Most large  
142 enterprises have already invested significant time and money in an e-business infrastructure and  
143 are reluctant to change the way they conduct electronic business. Furthermore, every company  
144 has different requirements for the information exchanged in a specific business process, such as  
145 procurement or supply-chain optimization. A standard business language must strike a difficult  
146 balance, adapting to the specific needs of a given company while remaining general enough to let  
147 different companies in different industries communicate with each other.

148 The UBL effort addresses this problem by building on the work of the electronic business XML  
149 (ebXML) initiative. UBL is organized as an OASIS Technical Committee to guarantee a rigorous,  
150 open process for the standardization of the XML business language. The development of UBL  
151 within OASIS also helps ensure a fit with other essential ebXML specifications.

152 This specification documents the rules and guidelines for the naming and design of XML  
153 components for the UBL library. It contains only rules that have been agreed on by the OASIS  
154 UBL Technical Committee. Consumers of the Naming and Design Rules Specification should  
155 consult previous UBL position papers that are available at [http://www.oasis-  
156 open.org/committees/ubl/ndrsc/](http://www.oasis-open.org/committees/ubl/ndrsc/). These provide a useful background to the development of the  
157 current rule set.

## 1.1 Audiences

159 This document has several primary and secondary targets that together constitute its intended  
160 audience. Our primary target audience is the members of the UBL Technical Committee.  
161 Specifically, the UBL Technical Committee will use the rules in this document to create normative  
162 form schemas for business transactions. Developers implementing ebXML Core Components  
163 may find the rules contained herein sufficiently useful to merit adoption as, or infusion into, their  
164 own approaches to ebXML Core Component based XML schema development. All other XML  
165 Schema developers may find the rules contained herein sufficiently useful to merit consideration  
166 for adoption as, or infusion into, their own approaches to XML schema development.

## 1.2 Scope

168 This specification conveys a normative set of XML schema design rules and naming conventions  
169 for the creation of business based XML schemas for business documents being exchanged  
170 between two parties using XML constructs defined in accordance with the ebXML Core  
171 Components Technical Specification.

## 172 1.3 Terminology and Notation

173 The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD**  
174 **NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as  
175 described in Internet Engineering Task Force (IETF) Request for Comments (RFC) 2119. Non-  
176 capitalized forms of these words are used in the regular English sense.

177 [Definition] – A formal definition of a term. Definitions are normative.

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

179 [Note] – Explanatory information. Notes are informative.

180 [RRRn] – Identification of a rule that requires conformance to ensure that an XML Schema is UBL  
181 conformant. The value RRR is a prefix to categorize the type of rule where the value of RRR is as  
182 defined in Table 1-1 and n (1..n) indicates the sequential number of the rule within its category. In  
183 order to ensure continuity across versions of the specification, rule numbers that are deleted in  
184 future versions will not be re-issued, and any new rules will be assigned the next higher number –  
185 regardless of location in the text. Future versions will contain an appendix that lists deleted rules  
186 and the reason for their deletion. Only rules and definitions are normative; all other text is  
187 explanatory.

188 **Table 1-1 Rule Prefix Token Value**

Rule Prefix Token	Value
ATD	Attribute Declaration
CDL	Code List
CTD	ComplexType Definition
CTN	ComplexType Naming Rules (CTN)
DOC	Documentation
ELD	Element Declaration
ELN	Element Naming
GNR	General Naming
GTD	General Type Definition
GXS	General XML Schema
IND	Instance Document
MDC	Modeling Constraints
NMC	Naming Constraints
NMS	Namespace
RED	Root Element Declaration
SSM	Schema Structure Modularity
VER	Versioning

189 **Bold** – The bolding of words is used to represent example names or parts of names taken from  
190 the library.

191 *Courier* – All words appearing in *courier* font are values, objects, and keywords.

192 *Italics* – All words appearing in italics, when not titles or used for emphasis, are special terms  
193 defined in Appendix B.

194 **Keywords** – keywords reflect concepts or constructs expressed in the language of their source  
195 standard. Keywords have been given an identifying prefix to reflect their source. The following  
196 prefixes are used:

197 `xsd:` – represents W3C XML Schema Definition Language. If a concept, the words will be in  
198 upper camel case, and if a construct, they will be in lower camel case.

199 `xsd:` – `complexType` represents an XSD construct  
200 `xsd:` – `SchemaExpression` represents a concept  
201 `ccts:` – represents ISO 15000-5 ebXML Core Components Technical Specification  
202 `ubl:` – represents the OASIS Universal Business Language  
203 The terms “W3C XML Schema” and “XSD” are used throughout this document. They are  
204 considered synonymous; both refer to XML Schemas that conform to Parts 1 and 2 of the W3C  
205 *XML Schema Definition Language (XSD)* Recommendations. See Appendix B for additional term  
206 definitions.

## 207 1.4 Guiding Principles

208 The UBL guiding principles encompass three areas:

- 209 ◆ General UBL guiding principles
- 210 ◆ Extensibility
- 211 ◆ Relationship to tools

### 212 1.4.1 Adherence to General UBL Guiding Principles

213 The UBL Technical Committee has approved a set of high-level guiding principles. These  
214 principles were adhered to during development of UBL NDR. These UBL guiding principles are:

- 215 ◆ Internet Use – UBL shall be straightforwardly usable over the Internet.
- 216 ◆ Interchange and Application Use – UBL is intended for interchange and application  
217 use.
- 218 ◆ Tool Use and Support – The design of UBL will not make any assumptions about  
219 sophisticated tools for creation, management, storage, or presentation being  
220 available. The lowest common denominator for tools is incredibly low (for example,  
221 Notepad) and the variety of tools used is staggering. We do not see this situation  
222 changing in the near term.
- 223 ◆ Legibility – UBL documents should be human-readable and reasonably clear.
- 224 ◆ Simplicity – The design of UBL must be as simple as possible (but no simpler).
- 225 ◆ 80/20 Rule – The design of UBL should provide the 20% of features that  
226 accommodate 80% of the needs.
- 227 ◆ Component Reuse – The design of UBL document types should contain as many  
228 common features as possible. The nature of e-commerce transactions is to pass  
229 along information that gets incorporated into the next transaction down the line. For  
230 example, a purchase order contains information that will be copied into the purchase  
231 order response. This forms the basis of our need for a core library of reusable

- 232 components. Reuse in this context is important, not only for the efficient development  
233 of software, but also for keeping audit trails.
- 234 ◆ Standardization – The number of ways to express the same information in a UBL  
235 document is to be kept as close to one as possible.
- 236 ◆ Domain Expertise – UBL will leverage expertise in a variety of domains through  
237 interaction with appropriate development efforts.
- 238 ◆ Customization and Maintenance – The design of UBL must facilitate customization  
239 and maintenance.
- 240 ◆ Context Sensitivity – The design of UBL must ensure that context-sensitive document  
241 types aren't precluded.
- 242 ◆ Prescriptiveness – UBL design will balance prescriptiveness in any single usage  
243 scenario with prescriptiveness across the breadth of usage scenarios supported.  
244 Having precise, tight content models and datatypes is a good thing (and for this  
245 reason, we might want to advocate the creation of more document type "flavors"  
246 rather than less). However, in an interchange format, it is often difficult to get the  
247 prescriptiveness that would be desired in any single usage scenario.
- 248 ◆ Content Orientation – Most UBL document types should be as "content-oriented" (as  
249 opposed to merely structural) as possible. Some document types, such as product  
250 catalogs, will likely have a place for structural material such as paragraphs, but these  
251 will be rare.
- 252 ◆ XML Technology – UBL design will avail itself of standard XML processing  
253 technology wherever possible (XML itself, XML Schema, XSLT, XPath, and so on).  
254 However, UBL will be cautious about basing decisions on "standards" (foundational  
255 or vocabulary) that are works in progress.
- 256 ◆ Relationship to Other Namespaces – UBL design will be cautious about making  
257 dependencies on other namespaces.
- 258 ◆ Legacy formats – UBL is not responsible for catering to legacy formats; companies  
259 (such as ERP vendors) can compete to come up with good solutions to permanent  
260 conversion. This is not to say that mappings to and from other XML dialects or non-  
261 XML legacy formats wouldn't be very valuable.

## 262 1.4.2 Design for Extensibility

263 UBL Naming and Design Rules 2.0 provides an extension mechanism to the meet the needs of  
264 customizers. This extension mechanism is embodied within 3.4 of the specification.

## 265 1.4.3 Relationship to Tools

266 The UBL NDR makes no assumptions on the availability or capabilities of tools to generate UBL  
267 conformant XSD schemas. In conformance with UBL guiding principles, the UBL NDR design  
268 process has scrupulously avoided establishing any naming or design rules that sub-optimize the  
269 UBL schemas in favor of tool generation. Additionally, in conformance with UBL guiding  
270 principles, the NDR is sufficiently rigorous to avoid requiring human judgment at schema  
271 generation time.



## 272 1.4.4 Choice of Schema Language

273 The W3C XML Schema Definition Language has become the generally accepted schema  
274 language that is experiencing the most widespread adoption. Although other schema languages  
275 exist that offer their own advantages and disadvantages, UBL has determined that the best  
276 approach for developing an international XML business standard is to base its work on W3C  
277 XSD. Consequently, all UBL schema design rules are based on the W3C XML Schema  
278 Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

279 By aligning with W3C specifications holding recommended status, UBL can ensure that its  
280 products and deliverables are well suited for use by the widest possible audience with the best  
281 availability of common support tools.

282

---

## 2 Relationship to ebXML Core Components

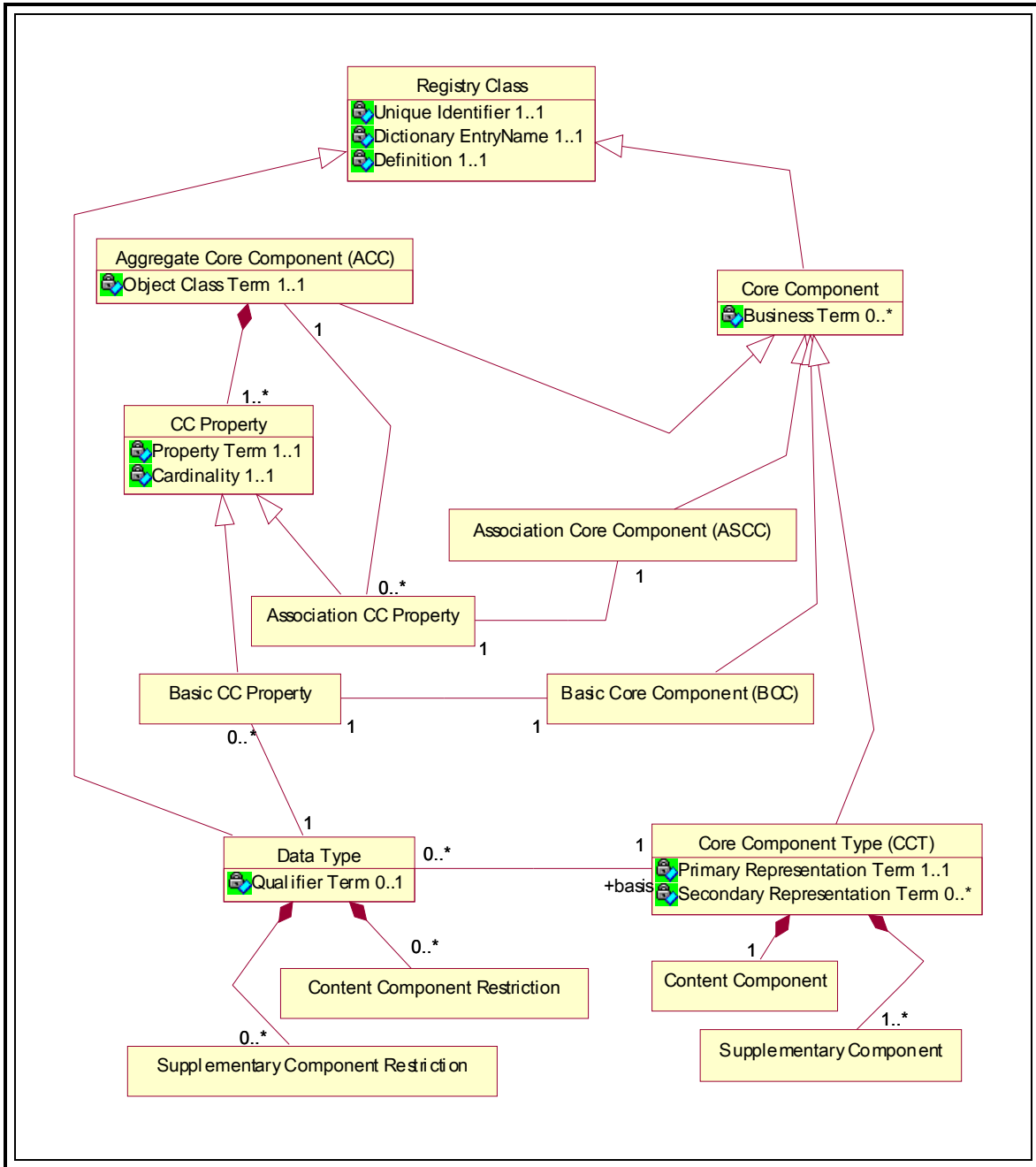
283 UBL employs the methodology and model described in *Core Components Technical*  
284 *Specification, ISO 15000-5* to build the UBL Component Library. The Core Components concept  
285 defines a new paradigm in the design and implementation of reusable syntactically neutral  
286 information building blocks. Syntax neutral Core Components are intended to form the basis of  
287 business information standardization efforts and to be realized in syntactically specific  
288 instantiations such as ANSI ASC X12, UN/EDIFACT, and XML representations such as UBL.

289 The essence of the Core Components specification is captured in context neutral and context  
290 specific building blocks. The context neutral components are defined as Core Components  
291 (`ccts:CoreComponents`). Context neutral `ccts:CoreComponents` are defined in CCTS as “A  
292 building block for the creation of a semantically correct and meaningful information exchange  
293 package. It contains only the information pieces necessary to describe a specific concept.” Figure  
294 2-1 illustrates the various pieces of the overall `ccts:CoreComponents` metamodel.

295 The context specific components are defined as Business Information Entities  
296 (`ccts:BusinessInformationEntities`). Context specific `ccts:Business`  
297 `InformationEntities` are defined in CCTS as “A piece of business data or a group of pieces  
298 of business data with a unique *Business Semantic* definition.” Figure 2-2 illustrates the various  
299 pieces of the overall `ccts:BusinessInformationEntity` metamodel and their relationship  
300 with the `ccts:CoreComponents` metamodel.

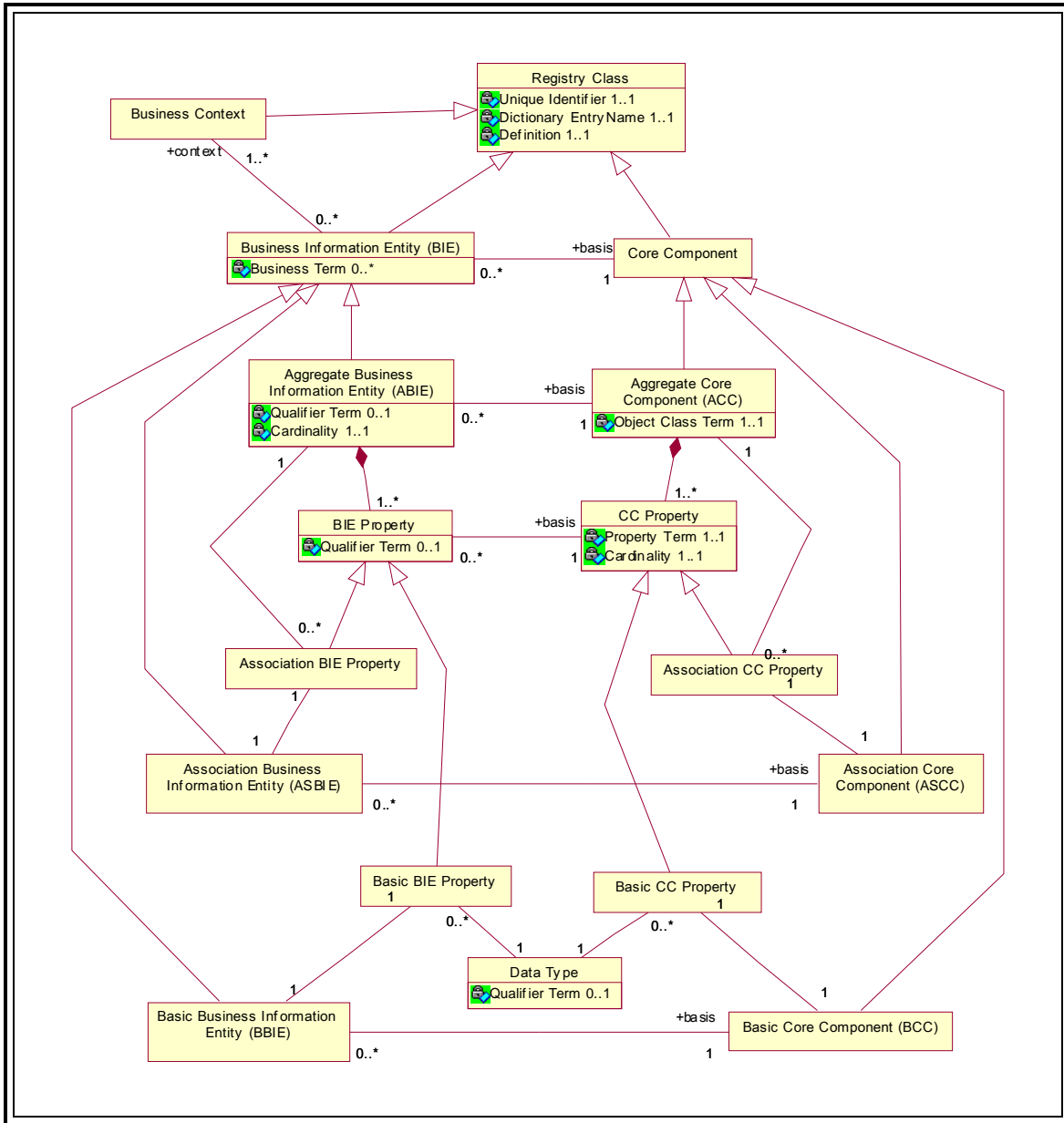
301 As shown in Figure 2-2, there are different types of `ccts:CoreComponents` and  
302 `ccts:BusinessInformationEntities`. Each type of `ccts:CoreComponent` and  
303 `ccts:BusinessInformationEntity` has specific relationships between and amongst the  
304 other components and entities. The context neutral `ccts:Core`  
305 `Components` are the linchpin that establishes the formal relationship between the various  
306 context-specific `ccts:BusinessInformationEntities`.

307 **Figure 2-1 Core Components and Datatypes**  
 308 **Metamodel**



309

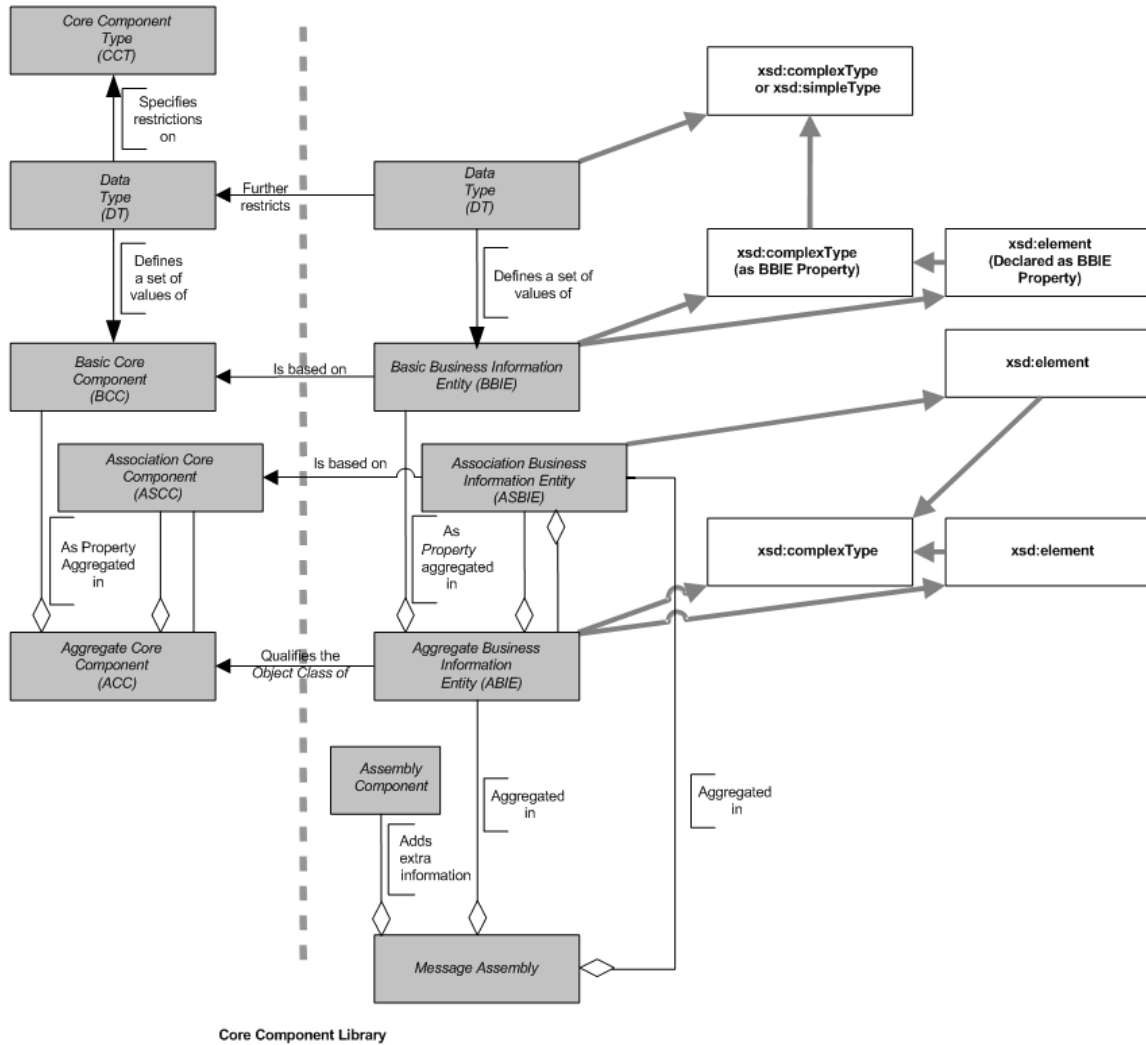
310 **Figure 2-2 Business Information Entities Basic Definition Model**



311

## 312 2.1 Mapping Business Information Entities to XSD

313 UBL consists of a library of `ccts:BusinessInformationEntities` (BIEs). In creating this  
 314 library, UBL has defined how each of the BIE components map to an XSD construct (See figure  
 315 2-3). In defining this mapping, UBL has analyzed the CCTS metamodel and determined the  
 316 optimal usage of XSD to express the various BIE components.



318  
 319 As stated above, a BIE can be a `ccts:AggregateBusinessInformationEntity` (ABIE),  
 320 a `ccts:BasicBusinessInformationEntity` (BBIE), or a  
 321 `ccts:AssociationBusinessInformationEntity` (ASBIE). In understanding the logic of  
 322 the UBL binding of BIEs to XSD expressions, it is important to understand the basic constructs of  
 323 the ABIEs and their relationships as shown in Figure 2-2.

324 Both Aggregate and Basic Business Information Entities must have a unique name (Dictionary  
 325 Entry Name). The ABIEs are treated as objects and are defined as `xsd:complexType`s. The  
 326 BBIEs are treated as attributes of the ABIE and are found in the content model of the ABIE as a  
 327 referenced `xsd:element`. The BBIEs are based on a reusable  
 328 `ccts:BasicBusinessInformationEntityProperty` (BBIE Property) which are  
 329 defined as `xsd:complexType`s.

330 A BBIE Property represents an *intrinsic* property of an ABIE. BBIE Properties are linked to a  
331 Datatype. UBL uses two types of Datatypes – unqualified, that are provided by the UN/CEFACT  
332 Unqualified Datatype (udt) schema module, and qualified datatypes that are defined by UBL.

333 UBL's use of the UN/CEFACT Unqualified Datatype schema module is primarily confined to its  
334 importation. It must not be assumed that UBL's adoption of the UDT schema module extends to  
335 any of the Advanced Technology Group's (ATG) rules that have a bearing on the use of the UDT.

336 The `ccts:UnqualifiedDatatypes` correspond to `ccts:RepresentationTerms`. The  
337 `ubl:QualifiedDatatypes` are derived from `ccts:UnqualifiedDatatypes` with  
338 restrictions to the allowed values or ranges of the corresponding `ccts:ContentComponent` or  
339 `ccts:SupplementaryComponent`.

340 CCTS defines an approved set of primary and secondary representation terms. However, these  
341 representation terms are simply naming conventions to identify the Datatype of an object, not  
342 actual constructs. These representation terms are in fact the basis for Datatypes as defined in the  
343 CCTS.

344 A `ccts:Datatype` “defines the set of valid values that can be used for a particular *Basic Core*  
345 *Component Property* or *Basic Business Information Entity Property Datatype*.” The  
346 `ccts:Datatypes` can be either unqualified—no restrictions applied—or qualified through the  
347 application of restrictions. The sum total of the datatypes is then instantiated as the basis for the  
348 various XSD simple and complex types defined in the UBL schemas. CCTS supports datatypes  
349 that are qualified, i.e. it enables users to define their own datatypes for their syntax neutral  
350 constructs. Thus `ccts:Datatypes` allow UBL to identify restrictions for elements when  
351 restrictions to the corresponding `ccts:ContentComponent` or `ccts:`  
352 `SupplementaryComponent` are required.

353 There are two kinds of Business Information Entity Properties - Basic and Association. A  
354 `ccts:AssociationBusinessInformationEntityProperty` (ASBIE Property) represents  
355 an *extrinsic* property – in other words an association from one ABIE instance to another ABIE  
356 instance. It is the ASBIE Property that expresses the relationship between ABIEs . Due to their  
357 unique extrinsic association role, ASBIEs are not defined as `xsd:complexType`s, rather they  
358 are either declared as elements that are then bound to the `xsd:complexType` of the associated  
359 ABIE ,or they are reclassified ABIEs.

360 As stated above, BBIEs define the intrinsic structure of an ABIE. These BBIEs are the “leaf” types  
361 in the system in that they contain no ASBIE Properties.

362 A BBIE must have a `ccts:CoreComponentType`. All `ccts:CoreComponentTypes` are low-  
363 level types, such as Identifiers and Dates. A `ccts:CoreComponentType` describes these low-  
364 level types for use by `ccts:CoreComponents`, and (in parallel) a `ccts:Datatype`,  
365 corresponding to that `ccts:CoreComponentType`, describes these low-level types for use by  
366 BBIEs. Every `ccts:CoreComponentType` has a single `ccts:ContentComponent` and one or  
367 more `ccts:SupplementaryComponents`. A `ccts:ContentComponent` is of some  
368 Primitive Type. All `ccts:CoreComponentTypes` and their corresponding content and  
369 supplementary components are pre-defined in the CCTS. UBL has developed an  
370 `xsd:SchemaModule` that defines each of the pre-defined `ccts:CoreComponentTypes` as an  
371 `xsd:complexType` or `xsd:simpleType` and declares `ccts:SupplementaryComponents`  
372 as an `xsd:attribute` or uses the predefined facets of the built-in `xsd:Datatype` for those  
373 that are used as the base expression for an `xsd:simpleType`. UBL continues to work with  
374 UN/CEFACT and the Open Applications Group to develop a single normative schema for  
375 representing `ccts:CoreComponentTypes`.

---

## 376 3 General XML Constructs

377 This chapter defines UBL rules related to general XML constructs to include:

378 Overall Schema Structure

379 Naming and Modeling Constraints

380 Reusability Scheme

381 Namespace Scheme

382 Versioning Scheme

383 Modularity Strategy

384 Annotation and Documentation Requirements

### 385 3.1 Overall Schema Structure

386 A key aspect of developing standards is to ensure consistency in their implementation. Therefore,  
387 it is essential to provide a mechanism that will guarantee that each occurrence of a UBL  
388 conformant schema will have the same look and feel.

389 [GXS1] UBL Schema, except in the case of extension, where the 'UBL Extensions' element is  
390 used, MUST conform to the following physical layout as applicable:

391 <!-- ===== XML Declaration===== -->

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

393 <!-- ===== Schema Header ===== -->

394 Document Name: < Document name as indicated in Section 3.6 >

395 Generated On: < Date schema was generated >

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

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

399 `xmlns:xsd`

400 Target namespace

401 Default namespace

402 CommonAggregateComponents

403 CommonBasicComponents

404 CoreComponentTypes

405 Unqualified Datatypes

406 Qualified Datatypes

407 Identifier Schemes  
408 Code Lists  
409 Attribute Declarations – elementFormDefault="qualified"  
410 attributeFormDefault="unqualified"  
411 Version Attribute  
412 <!-- ===== Imports ===== -->  
413 CommonAggregateComponents schema module  
414 CommonBasicComponents schema module  
415 Unqualified Types schema module  
416 Qualified Types schema module  
417 <!-- ===== Root Element ===== -->  
418 Root Element Declaration  
419 Root Element Type Definition  
420 <!-- ===== Element Declarations ===== -->  
421 alphabetized order  
422 <!-- ===== Type Definitions ===== -->  
423 All type definitions segregated by basic and aggregates as follows  
424 <!-- ===== Aggregate Business Information Entity Type Definitions ===== -->  
425 alphabetized order of ccts:AggregateBusinessInformationEntity xsd:TypeDefinitions  
426 <!-- ===== Basic Business Information Entity Type Definitions ===== -->  
427 alphabetized order of ccts:BasicBusinessInformationEntities  
428 <!-- ===== Copyright Notice ===== -->  
429 Required OASIS full copyright notice.

### 430 3.1.1 Element declarations within document schemas

431 [Definition] Document schema –  
432 The overarching schema within a specific namespace that conveys the business document  
433 functionality of that namespace. The document schema declares a target namespace and is  
434 likely to xsd:include internal schema modules or xsd:import external schema modules. Each  
435 namespace will have one, and only one, document schema.

436 In order to facilitate the management and reuse of UBL constructs, all global elements, excluding  
437 the root element of the document schema, must reside in either the Common Aggregate  
438 Components (CAC) or Common Basic Components (CBC) schema modules.

#### 439 3.1.1.1 Root Element

440 UBL has chosen a global element approach. Inside a UBL document schema only a single global  
441 element is declared. Because all UBL instance documents conform to a UBL document schema,



442 the single global element declared in that document schema will be the root element of the  
443 instance.

444 [RED2] The root element MUST be the only global element declared in document schemas.

445

## 446 3.2 Naming and Modeling Constraints

447 A key aspect of UBL is to base its work on process modeling and data analysis as precursors to  
448 developing the UBL library. In determining how best to affect this work, several constraints have  
449 been identified that directly impact the process modeling and data analysis, as well as the  
450 resultant UBL Schema.

### 451 3.2.1 Naming Constraints

452 A primary aspect of the UBL library documentation is its spreadsheet models. The entries in  
453 these spreadsheet models fully define the constructs available for use in UBL business  
454 documents. These spreadsheet entries contain fully conformant CCTS dictionary entry names as  
455 well as truncated UBL XML element names developed in conformance with the rules in section 4.  
456 The dictionary entry name ties the information to its standardized semantics, while the name of  
457 the corresponding XML element is only shorthand for this full name. The rules for element naming  
458 and dictionary entry naming are different.

459 [NMC1] Each dictionary entry name MUST define one and only one fully qualified path (FQP)  
460 for an element or attribute.

461 The fully qualified path anchors the use of that construct to a particular location in a business  
462 message. The definition of the construct identifies any semantic dependencies that the FQP has  
463 on other elements and attributes within the UBL library that are not otherwise enforced or made  
464 explicit in its structural definition.

### 465 3.2.2 Modeling Constraints

466 In keeping with UBL guiding principles, modeling constraints are limited to those necessary to  
467 ensure consistency in development of the UBL library.

#### 468 3.2.2.2 Defining Classes

469 UBL is based on instantiating ebXML `ccts:BusinessInformationEntities` (BIEs). UBL  
470 models and the XML expressions of those models are class driven. Specifically, the UBL library  
471 defines classes for each `ccts:AggregateBusinessInformationEntity` (ABIE) and the  
472 UBL schemas instantiate those classes. The attributes of those classes consist of  
473 `ccts:BasicBusinessInformationEntities` (BBIEs).

#### 474 3.2.2.3 Core Component Types

475 Each BBIE has an associated `ccts:CoreComponentType`. The CCTS specifies an approved  
476 set of `ccts:CoreComponentTypes`. To ensure conformance, UBL is limited to using this  
477 approved set.

478 [MDC1] UBL Libraries and Schemas MUST only use ebXML Core Component approved  
479 ccts:CoreComponentTypes, except in the case of extension, where the  
480 'UBLExtensions' element is used.

481 Customization is a key aspect of UBL's reusability across business verticals. The UBL rules have  
482 been developed in recognition of the need to support customizations. Specific UBL customization  
483 rules are detailed in the UBL customization guidelines.

#### 484 3.2.2.4 Mixed Content

485 UBL documents are designed to effect data-centric electronic commerce. Including mixed content  
486 in business documents is undesirable because business transactions are based on exchange of  
487 discrete pieces of data that must be clearly unambiguous. The white space aspects of mixed  
488 content make processing unnecessarily difficult and add a layer of complexity not desirable in  
489 business exchanges.

490 [MDC2] Mixed content MUST NOT be used except where contained in an  
491 xsd:documentation element.

### 492 3.3 Reusability Scheme

493 The effective management of the UBL library requires that all element declarations are unique  
494 across the breadth of the UBL library. Consequently, UBL elements are declared globally.

#### 495 3.3.1.5 Reusable Elements

496 UBL elements are global and qualified. Hence in the example below, the <Address> element is  
497 directly reusable as a modular component and some software can be used without modification.

#### 498 Example

```
499 <xsd:element name="Party" type="PartyType"/>
500 <xsd:complexType name="PartyType">
501   <xsd:annotation>
502     <!--Documentation goes here -->
503   </xsd:annotation>
504   <xsd:sequence>
505     <xsd:element ref="cbc:MarkCareIndicator" minOccurs="0" maxOccurs="1">
506       ...
507     </xsd:element>
508     <xsd:element ref="cbc:MarkAttentionIndicator" minOccurs="0" maxOccurs="1">
509       ...
510     </xsd:element>
511     <xsd:element ref="PartyIdentification" minOccurs="0" maxOccurs="unbounded">
512       ...
513     </xsd:element>
514     <xsd:element ref="PartyName" minOccurs="0" maxOccurs="1">
515       ...
516     </xsd:element>
517     <xsd:element ref="Address" minOccurs="0" maxOccurs="1">
518       ...
519     </xsd:element>
520   ...
521   </xsd:sequence>
522 </xsd:complexType>
523 <xsd:element name="Address" type="AddressType"/>
524 <xsd:complexType name="AddressType">
525   ...
526 </xsd:sequence>
```

527  
528  
529  
530  
531  
532  
533  
534  
535

```
<xsd:element ref="cbc:CityName" minOccurs="0" maxOccurs="1">
  ...
</xsd:element>
<xsd:element ref="cbc:PostalZone" minOccurs="0" maxOccurs="1">
  ...
</xsd:element>
  ...
</xsd:sequence>
</xsd:complexType>
```

536 Software written to work with UBL's standard library will work with new assemblies of the same  
537 components since global elements will remain consistent and unchanged. The globally declared  
538 <Address> element is fully reusable without regard to the reusability of types and provides a  
539 solid mechanism for ensuring that extensions to the UBL core library will provide consistency and  
540 semantic clarity regardless of its placement within a particular type.

541 [ELD2] All element declarations MUST be global

## 542 3.4 Extension Scheme

543 There is a recognized requirement that some organizations are required by law to send additional  
544 information not covered by the UBL document structure, thus requiring an extension to the UBL  
545 message. The `xsd:any` construct is seen as the most efficient way to implement this  
546 requirement.

547 In general, UBL restricts the use of `xsd:any` because this feature permits the introduction of  
548 potentially unknown elements into an XML instance. However, limiting its use to a single,  
549 predefined element mitigates this risk. Since it is a priority that there can be meaningful validation  
550 of the UBL document instances the value of the `xsd:processContents` attribute of the element  
551 must be set to "skip", thereby removing the potential for errors in the validation layer. There is  
552 cardinality restriction in the case of extension.

553 [GXS14] The `xsd:any` element MUST NOT be used except within the  
554 'ExtensionContentType' type definition, and with `xsd:processContents= "skip"`  
555 for non-UBL namespaces.

556 The following rules apply in the order below.  
557

558 [ELD12] The 'UBL Extensions' element MUST be declared as the first child of the document  
559 element with `xsd:minOccurs="0"`.

560

561 [ELD13] The 'UBLProfileID' element MUST be declared immediately following the 'UBL  
562 Extensions' element with `xsd:minOccurs="0"`.

563

564 [ELD14] The 'UBLSubsetID' element MUST be declared immediately following the  
565 'UBLProfileID' element with `xsd:minOccurs="0"`.

566

## 567 3.5 Namespace Scheme

568 The concept of XML namespaces is defined in the W3C XML namespaces technical  
569 specification. The use of XML namespace is specified in the W3C XML Schema (XSD)  
570 Recommendation. A namespace is declared in the root element of a Schema using a namespace  
571 identifier. Namespace declarations can also identify an associated prefix—shorthand identifier—  
572 that allows for compression of the namespace name. For each UBL namespace, a normative  
573 token is defined as its prefix. These tokens are defined in the versioning scheme section.

### 574 3.5.1 Declaring Namespaces

575 Neither XML 1.0 nor XSD require the use of Namespaces. However the use of namespaces is  
576 essential to managing the complex UBL library. UBL will use UBL-defined schemas (created by  
577 UBL) and UBL-used schemas (created by external activities) and both require a consistent  
578 approach to namespace declarations.

579 [NMS1] Every UBL-defined –or -used schema module, except internal schema modules,  
580 MUST have a namespace declared using the `xsd:targetNamespace` attribute.

581 Each UBL schema module consists of a logical grouping of lower level artifacts that together  
582 comprise an association that will be able to be used in a variety of UBL schemas. These schema  
583 modules are grouped into a schema set. Each schema set is assigned a namespace that  
584 identifies that group of schema modules. As constructs are changed, new versions will be  
585 created. The schema set is the versioned entity, all schema modules within that package are of  
586 the same version, and each version has a unique namespace.

587 [Definition] Schema Set –

588 A collection of schema instances that together comprise the names in a specific UBL  
589 namespace.

590 Schema validation ensures that an instance conforms to its declared schema. There should never  
591 be two (different) schemas with the same namespace Uniform Resource Identifier (URI). In  
592 keeping with Rule NMS1, each UBL schema module will be part of a versioned namespace.

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

595 UBL's extension methodology encourages a wide variety in the number of schema modules that  
596 are created as derivations from UBL schema modules. Clarity and consistency requires that  
597 customized schema not be confused with those developed by UBL.

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

### 599 3.5.2 Namespace Uniform Resource Identifiers

600 A UBL namespace name must be a URI reference that conforms to RFC 2396. UBL has adopted  
601 the Uniform Resource Name (URN) scheme as the standard for URIs for UBLnamespaces, in  
602 conformance with IETF's RFC 3121, as defined in this next section.

603 Rule NMS2 requires separate namespaces for each UBL schema set. The UBL namespace rules  
604 differentiate between committee draft and OASIS Standard status. For each schema holding draft  
605 status, a UBL namespace must be declared and named.

606 [NMS4] The namespace names for UBL Schemas holding committee draft status MUST be of  
607 the form:  
608 urn:oasis:names:tc:ubl:schema:<subtype>:<document-id>

609 The format for `document-id` is found in the next section.

610 For each UBL schema holding OASIS Standard status, a UBL namespace must be declared and  
611 named using the same notation, but with the value 'specification' replacing the value 'tc'.

612 [NMS5] The namespace names for UBL Schemas holding OASIS Standard status MUST be  
613 of the form:  
614 urn:oasis:names:specification:ubl:schema:<subtype>:<document-  
615 id>  
616

### 617 3.5.3 Schema Location

618 UBL schemas use a URN namespace scheme. In contrast, schema locations are typically  
619 defined as a Uniform Resource Locator (URL). UBL schemas must be available both at design  
620 time and run time. As such, the UBL schema locations will differ from the UBL namespace  
621 declarations. UBL, as an OASIS TC, will utilize an OASIS URL for hosting UBL schemas. UBL  
622 will use the committee directory <http://www.oasis-open.org/committees/ubl/schema/>.

### 623 3.5.4 Persistence

624 A key differentiator in selecting URNs to define UBL namespaces is URN persistence. UBL  
625 namespaces must never violate this functionality by subsequently changing once it has been  
626 declared. Conversely, changes to a schema may result in a new namespace declaration. Thus a  
627 published schema version and its namespace association will always be inviolate.

628 [NMS6] UBL published namespaces MUST never be changed.

## 629 3.6 Versioning Scheme

630 UBL has adopted a two-layer versioning scheme. Major version information is captured within the  
631 namespace name of each UBL schema module while combined major and minor version  
632 information is captured within the `xsd:version` attribute of the `xsd:schema` element.

633 UBL namespaces conform to the OASIS namespace rules defined in RFC 3121. The last field of  
634 the namespace name is called `document-id`. UBL has decided to include versioning  
635 information as part of the `document-id` component of the namespace. Only major version  
636 information will be captured within the `document-id`. The `major` field has an optional  
637 `revision` extension which can be used for draft schemas. For example, the namespace URI for  
638 the draft Invoice domain has this form:

639 urn:oasis:names:tc:ubl:schema:xsd:Invoice-<major>[.<revision>]

640 The *major-version* field is “1” for the first release of a namespace. Subsequent major releases  
641 increment the value by 1. For example, the first namespace URI for the first major release of the  
642 Invoice document has the form:

643 urn:oasis:names:tc:ubl:schema:xsd:Invoice-1

644 The second major release will have a URI of the form:

645 urn:oasis:names:tc:ubl:schema:xsd:Invoice-2

646 In general, the namespace URI for every major release of the Invoice domain has the form:

647 urn:oasis:names:tc:ubl:schema:xsd:Invoice:-<major-number>[.<revision>]

648

649 [VER1] Every UBL Schema and schema module major version committee draft MUST have  
650 an RFC 3121 document-id of the form

651 <name>-<major>[.<revision>]

652

653 [VER11] Every UBL Schema and schema module major version committee draft MUST  
654 capture its version number in the `xsd:version` attribute of the `xsd:schema`  
655 element in the form

656 <major>.0[.<revision>]

657

658 [VER2] Every UBL Schema and schema module major version OASIS Standard MUST have  
659 an RFC 3121 document-id of the form

660 <name>-<major>

661

662 [VER12] Every UBL Schema and schema module major version OASIS Standard MUST  
663 capture its version number in the `xsd:version` attribute of the `xsd:schema`  
664 element in the form

665 <major>.0

666 For each document produced by the TC, the TC will determine the value of the <name> variable.  
667 In UBL, the major-version field must be changed in a release that breaks compatibility with the  
668 previous release of that namespace. If a change does not break compatibility then only the minor  
669 version need change. Subsequent minor releases begin with minor-version 1.

670 Example

671 The namespace URI for the first minor release of the Invoice domain has this form:

672

673 urn:oasis:names:tc:ubl:schema:xsd:Invoice-<major>

674

675 The value of the `xsd:schema` `xsd:version` attribute for the first minor release of the Invoice  
676 domain has this form:

677

678 <major>.1

679

680 [VER3] Every minor version release of a UBL schema or schema module committee draft  
681 MUST have an RFC 3121 document-id of the form

682 <name>-<major>[.<revision>]

683

685 [VER13] Every minor version release of a UBL schema or schema module committee draft  
686 MUST capture its version information in the `xsd:version` attribute in the form  
687 `<major>.<non-zero>[.<revision>]`

~~688~~

691 [VER4] Every minor version release of a UBL schema or schema module OASIS Standard  
692 MUST have an RFC 3121 document-id of the form  
693 `<name>-<major>`

~~694~~

696 [VER14] Every minor version release of a UBL schema or schema module OASIS Standard  
697 MUST capture its version information in the `xsd:version` attribute in the form  
698 `<major>.<non-zero>`

699 Once a schema version is assigned a namespace, that schema version and that namespace will  
700 be associated in perpetuity. However, because minor schema versions will retain the major  
701 version namespace, this is not a one-to-one relationship.

702 [VER5] For UBL Minor version changes the namespace name MUST not change,

703 UBL is composed of a number of interdependent namespaces. For instance, namespaces whose  
704 URI's start with `urn:oasis:names:tc:ubl:schema:xsd:Invoice-*` are dependent upon  
705 the common basic and aggregate namespaces, whose URI's have the form  
706 `urn:oasis:names:tc:ubl:schema:xsd:CommonBasicComponents-*` and  
707 `urn:oasis:names:tc:ubl:schema:xsd:CommonAggregateComponents-*` respectively.  
708 If either of the common namespaces requires a major version change then its namespace URI  
709 must change. If its namespace URI changes then any schema that imports the *new version* of the  
710 namespace must also change (to update the namespace declaration). And since this would  
711 require a major version change to the importing schema, its namespace URI in turn must change.  
712 The outcome is twofold:

713 ◆ There should never be ambiguity at the point of reference in a namespace  
714 declaration or version identification. A dependent schema imports precisely the  
715 version of the namespace that is needed. The dependent schema never needs to  
716 account for the possibility that the imported namespace can change.

717 ◆ When a dependent schema is upgraded to import a new version of a schema, the  
718 dependent schema's version must change.

719 Minor version changes, however, would not require changes to the namespace URI of any  
720 schemas. Because of this, semantic compatibility across minor versions (as well as major  
721 versions) is essential. Semantic compatibility in this sense pertains to preserving the business  
722 function.

723 [VER10] UBL Schema and schema module minor version changes MUST not break semantic  
724 compatibility with prior versions.

725 Version numbers are based on a logical progression. All major and minor version numbers will be  
726 based on positive integers. Version numbers always increment positively by one.

727 [VER6] Every UBL Schema and schema module major version number MUST be a  
728 sequentially assigned, incremental number greater than zero.

729

730 [VER7] Every UBL Schema and schema module minor version number MUST be a  
731 sequentially assigned, incremental non-negative integer.

732 UBL version information will also be captured in instances of UBL document schemas via a  
733 ubl:UBLVersionID element.

734 [VER15] Every UBL document schema MUST declare an optional element named  
735 "UBLVersionID" immediately following the optional 'UBL Extensions' element.

## 736 3.7 Modularity Strategy

737 There are many possible mappings of XML schema constructs to namespaces and to files. In  
738 addition to the logical taming of complexity that namespaces provide, dividing the physical  
739 realization of schema into multiple files—schema modules—provides a mechanism whereby  
740 reusable components can be imported as needed without the need to import overly complex  
741 complete schema.

742 [SSM1] UBL Schema expressions MAY be split into multiple schema modules.

743 [Definition] schema module –  
744 A schema document containing type definitions and element declarations intended to be  
745 reused in multiple schemas.

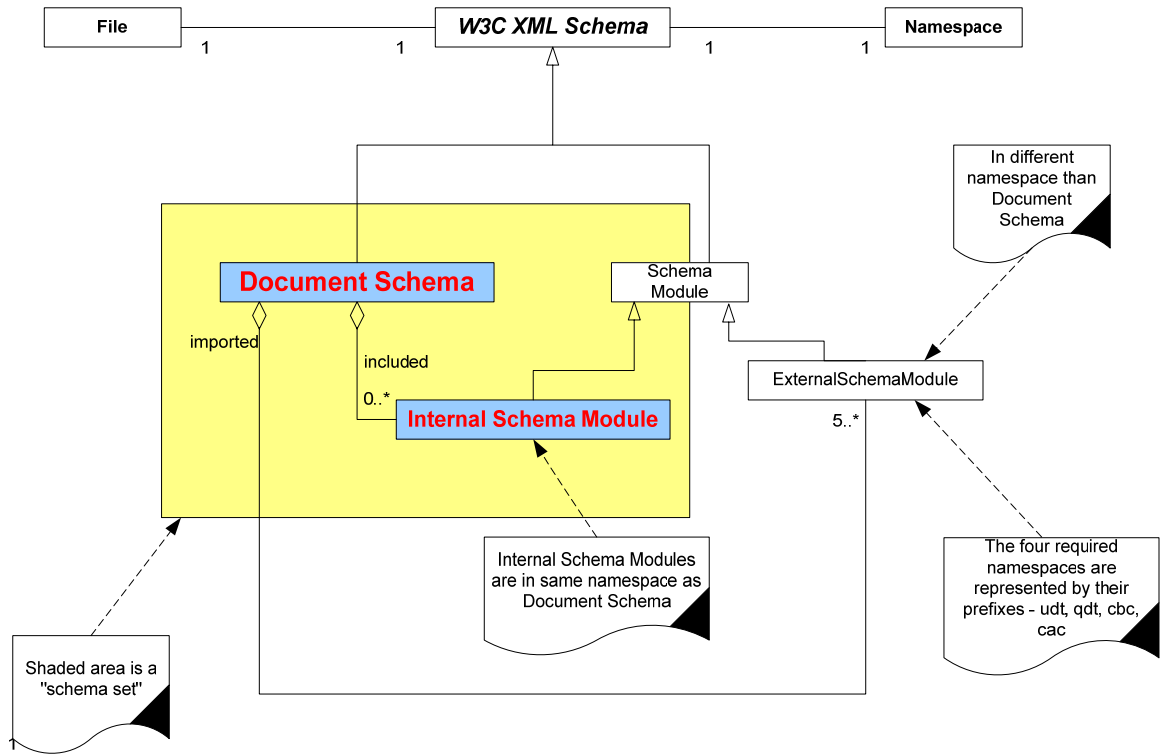
### 746 3.7.1 UBL Modularity Model

747 UBL relies extensively on modularity in schema design. There is no single UBL root schema.  
748 Rather, there are a number of UBL document schemas, each of which expresses a separate  
749 business function. The UBL modularity approach is structured so that users can reuse individual  
750 document schemas without having to import the entire UBL document schema library.  
751 Additionally, a document schema can import individual modules without having to import all UBL  
752 schema modules. Each document schema will define its own dependencies. The UBL schema  
753 modularity model ensures that logical associations exist between document and internal schema  
754 modules and that individual modules can be reused to the maximum extent possible. This is  
755 accomplished through the use of document and internal schema modules as shown in Figure 3-1.

756 If the contents of a namespace are small enough then they can be completely specified within a  
757 single schema.



758 **Figure 3-1 UBL Schema Modularity Model**



udt = Unqualified Datatype, qdt = Qualified Datatype, cbc = Common Basic Components, cac = Common Aggregate Components,

759  
760

761 Figure 3-1 shows the one-to-one correspondence between document schemas and namespaces.  
762 It also shows the one-to-one correspondence between files and schema modules. As shown in  
763 figure 3-1, there are two types of schema in the UBL library – document schema and schema  
764 modules. Document schemas are always in their own namespace. Schema modules may be in a  
765 document schema namespace as in the case of internal schema modules, or in a separate  
766 namespace as in the `ubl:qdt`, `ubl:cbc` and `ubl:cac` schema modules. Both types of schema  
767 modules are conformant with W3C XSD.

768 A namespace is a collection of semantically related elements, types and attributes. For larger  
769 namespaces, schema modules – internal schema modules – may be defined. UBL document  
770 schemas may have zero or more internal modules that they include. The document schema for a  
771 namespace then includes those internal modules.

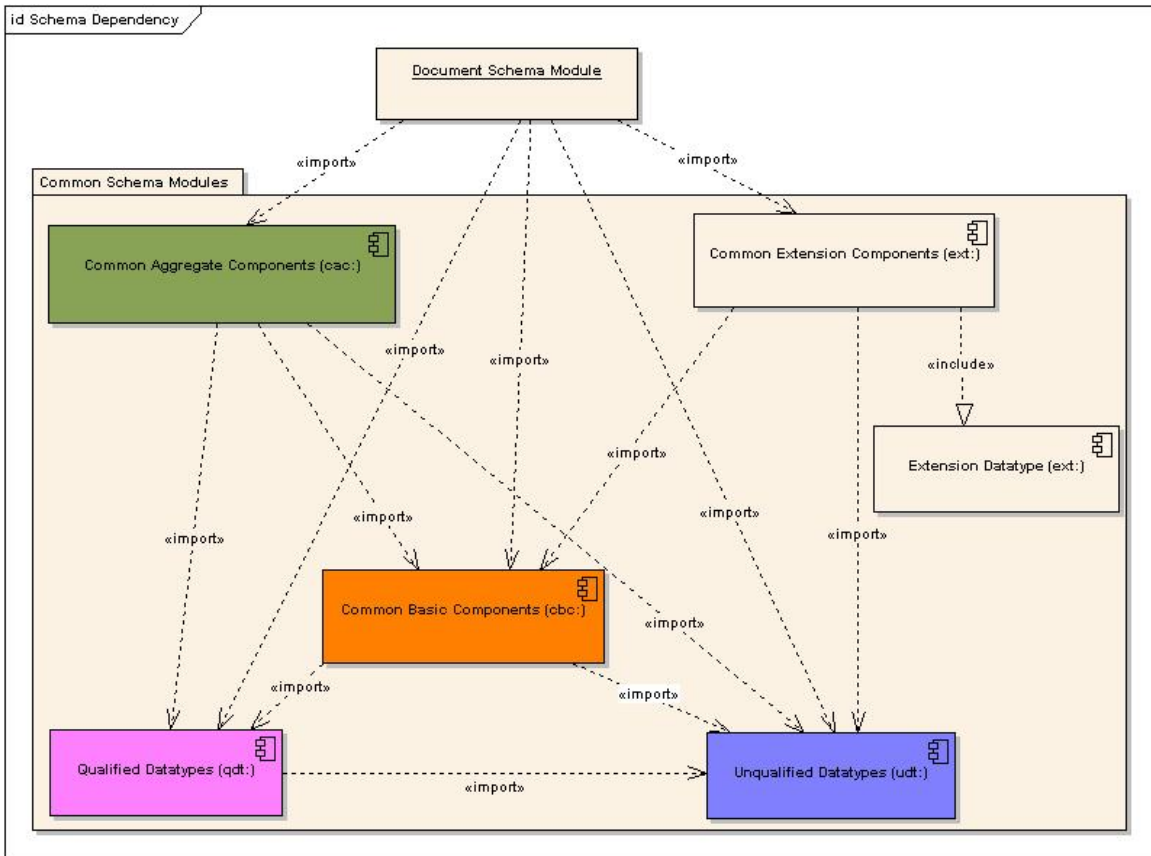
772

[Definition] Internal schema module –

773

A schema that is part of a schema set within a specific namespace.

774 **Figure 3-2 Schema Modules**



775

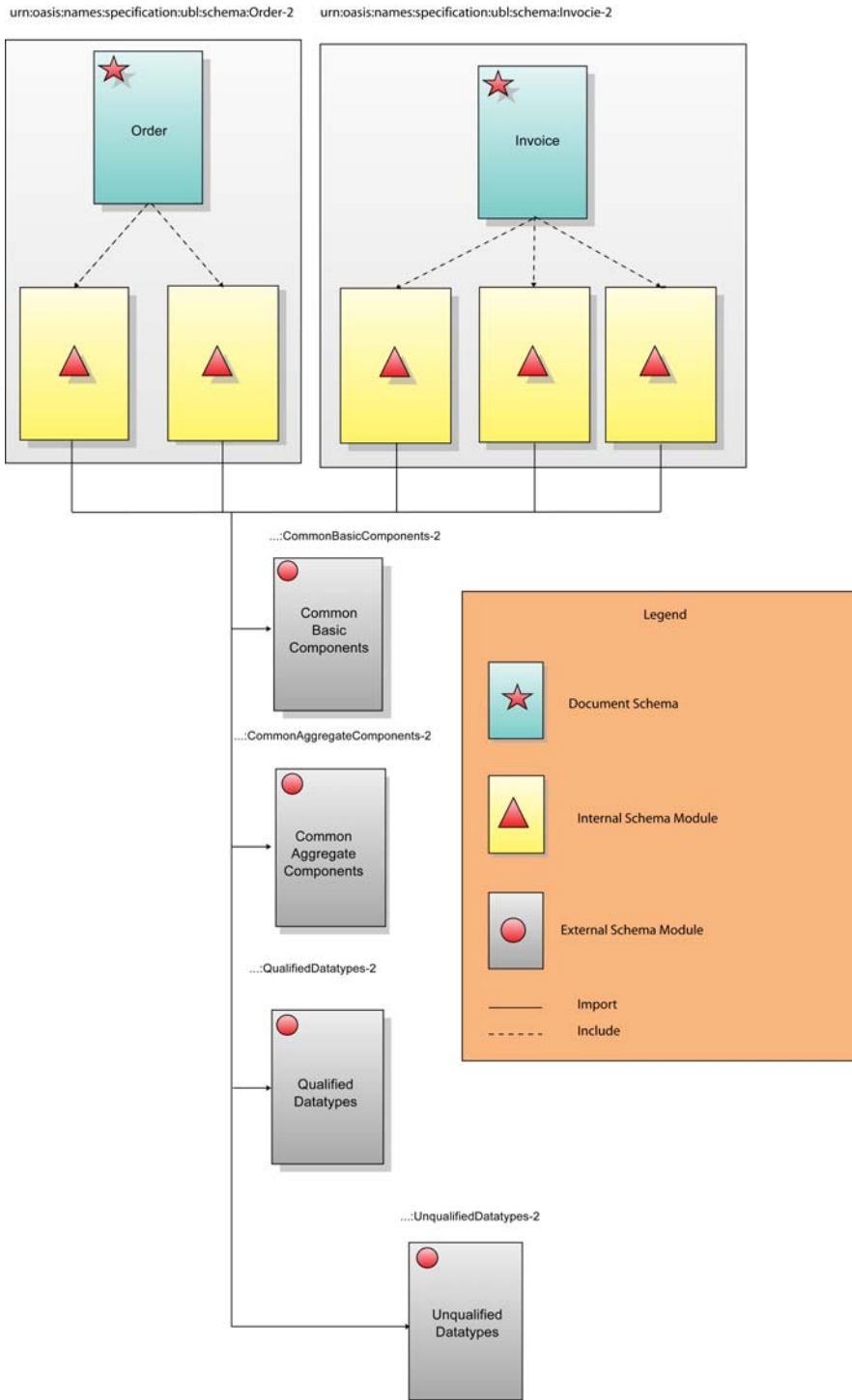
776

777 Another way to visualize the structure is by example. Figure 3-2 depicts instances of the various  
778 schema modules from the previous diagram.

779 Figure 3-3 shows how the order and invoice document schemas import the  
780 "CommonAggregateComponents Schema Module" and "CommonBasicComponents Schema  
781 Module" external schema modules. It also shows how the order document schema includes  
782 various internal modules – modules local to that namespace. The clear boxes show how the  
783 various schema modules are grouped into namespaces.

784 Any UBL schema module, be it a document schema or an internal module, may import other  
785 document schemas from other namespaces.

786 **Figure 3-3 Order and Invoice Schema Import of Common Component Schema**  
 787 **Modules**



788

### 789 3.7.1.6 Limitations on Import

790 If two namespaces are mutually dependent then clearly, importing one will cause the other to be  
791 imported as well. For this reason there must not exist circular dependencies between UBL  
792 schema modules. By extension, there must not exist circular dependencies between  
793 namespaces. A namespace “A” dependent upon type definitions or element declaration defined in  
794 another namespace “B” must import “B’s” document schema.

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

798 To ensure there is no ambiguity in understanding this rule, an additional rule is necessary to  
799 address potentially circular dependencies as well – schema A must not import internal schema  
800 modules of schema B.

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

### 804 3.7.2 Internal and External Schema Modules

805 As illustrated in Figure 3-1 and 3-2 UBL schema modules will be either internal or external  
806 schema modules.

### 807 3.7.3 Internal Schema Modules

808 UBL internal schema modules do not declare a target namespace, but instead reside in the  
809 namespace of their parent schema. All internal schema modules will be accessed using  
810 `xsd:include`.

811 [SSM6] All UBL internal schema modules MUST be in the same namespace as their  
812 corresponding document schema.

813 UBL internal schema modules will necessarily have semantically meaningful names. Internal  
814 schema module names will identify the parent schema module, the internal schema module  
815 function, and the schema module itself.

816 [SSM7] Each UBL internal schema module MUST be named  
817 {ParentSchemaModuleName}{InternalSchemaModuleFunction}{schema  
818 module}

### 819 3.7.4 External Schema Modules

820 UBL is dedicated to maximizing reuse. As the complex types and global element declarations will  
821 be reused in multiple UBL schemas, a logical modularity approach is to create UBL schema  
822 modules based on collections of reusable types and elements.

823 [SSM8] A UBL schema module MAY be created for reusable components.

824 As identified in rule SSM2, UBL will create external schema modules. These external schema  
825 modules will be based on logical groupings of contents. At a minimum, UBL schema modules will  
826 be comprised of:

827 UBL CommonAggregateComponents

828 UBL CommonBasicComponents

829 UBL Qualified Datatypes

830 In addition UBL will use the following schema modules provided by UN/CEFACT.

831 CCTS Core Component Types

832 CCTS Unqualified Datatypes

833 UN/CEFACT Code Lists

834 Furthermore, where extensions are used an extension schema module must be provided. This  
835 schema module must be named:

836 CommonExtensionComponents

837 This schema module must not import UBL-defined external schema modules.

838 [SSM21] The UBL extensions schema module MUST be identified as  
839 *CommonExtensionComponents* in the document name within the schema header.

#### 840 3.7.4.7 UBL Common Aggregate Components Schema Module

841 The UBL library will also contain a wide variety of *ccts:AggregateBusiness*  
842 *InformationEntities* (ABIEs). As defined in rule CTD1, each of these ABIEs will be defined  
843 as an *xsd:complexType*. Although some of these complex types may be used in only one UBL  
844 Schema, many will be reused in multiple UBL schema modules. An aggregation of all of the ABIE  
845 *xsd:complexType* definitions that are used in multiple UBL schema modules into a single  
846 schema module of common aggregate types will provide for maximum ease of reuse.

847 [SSM9] A schema module defining all UBL Common Aggregate Components MUST be  
848 created.

849 The normative name for this *xsd:ComplexType* schema module will be based on its ABIE  
850 content.

851 [SSM10] The UBL Common Aggregate Components schema module MUST be identified as  
852 *CommonAggregateComponents* in the document name within the schema header.

853 Example

854 Document Name: CommonAggregateComponents

##### 855 3.7.4.7.1 UBL CommonAggregateComponents Schema Module Namespace

856 In keeping with the overall UBL namespace approach, a singular namespace must be created for  
857 storing the *ubl:CommonAggregateComponents* schema module.

858 [NMS7] The *ubl:CommonAggregateComponents* schema module MUST reside in its own  
859 namespace.

860 To ensure consistency in expressing this module, a normative token that will be used consistently  
861 in all UBL Schemas must be defined.

862 [NMS8] The `ubl:CommonAggregateComponents` schema module namespace MUST be  
863 represented by the namespace prefix “cac” when referenced in other schemas.

### 864 3.7.4.8 UBL CommonBasicComponents Schema Module

865 The UBL library will contain a wide variety of `ccts:BasicBusinessInformation`  
866 `Entities` (BBIEs). These BBIEs are based on  
867 `ccts:BasicBusinessInformationEntityProperties` (BBIE Properties). BBIE  
868 Properties are reusable in multiple BBIEs. As defined in rule CTD25, each of these BBIE  
869 Properties is defined as an `xsd:complexType`. Although some of these complex types may be  
870 used in only one UBL Schema, many will be reused in multiple UBL schema modules. To  
871 maximize reuse and standardization, all of the BBIE properties `xsd:ComplexType` definitions  
872 that are used in multiple UBL schema modules will be aggregated into a single schema module of  
873 common basic types.

874 [SSM11] A schema module defining all UBL Common Basic Components MUST be created.

875 The normative name for this schema module will be based on its BBIE property  
876 `xsd:ComplexType` content.

877 [SSM12] The UBL Common Basic Components schema module MUST be identified as  
878 *CommonBasicComponents* in the document name within the schema header.

#### 879 3.7.4.8.1 UBL CommonBasicComponents Schema Module Namespace

880 In keeping with the overall UBL namespace approach, a singular namespace must be created for  
881 storing the `ubl:CommonBasicComponents` schema module.

882 [NMS9] The `ubl:CommonBasicComponents` schema module MUST reside in its own  
883 namespace.

884 To ensure consistency in expressing the `ubl:CommonBasicComponents` schema module, a  
885 normative token that will be used consistently in all UBL Schema must be defined.

886 [NMS10] The `ubl:CommonBasicComponents` schema module namespace MUST be  
887 represented by the namespace prefix “cbc” when referenced in other schemas.

### 888 3.7.4.9 CCTS CoreComponentType Schema Module

889 The CCTS defines an authorized set of Core Component Types (`ccts:Core`  
890 `ComponentTypes`) that convey content and supplementary information related to exchanged  
891 data. As the basis for all higher level CCTS models, the `ccts:Core`  
892 `ComponentTypes` are reusable in every UBL schema. An external schema module consisting of  
893 a complex type definition for each `ccts:CoreComponentType` is essential to maximize  
894 reusability. UBL uses the `ccts:CoreComponentType` schema module provided by the  
895 UN/CEFACT CCTS Datatypes Schema Modules

896 The CCTS defines an authorized set of primary and secondary Representation Terms  
897 (`ccts:RepresentationTerms`) that describes the form of every `ccts:Business`

898 InformationEntity. These `ccts:RepresentationTerms` are instantiated in the form of  
899 datatypes that are reusable in every UBL schema. The `ccts:Datatype` defines the set of valid  
900 values that can be used for its associated `ccts:BasicBusiness`  
901 InformationEntity Property. These datatypes may be qualified or unqualified, that is to  
902 say restricted or unrestricted. We refer to these as `ccts:Unqualified`  
903 Datatypes (even though they are technically `ccts:Datatypes`) or  
904 `ubl:QualifiedDatatypes`.

#### 905 *3.7.4.9.1 CCTS Unqualified Datatypes Schema Module*

906 UBL has adopted the UN/CEFACT Unqualified Datatype schema module. This includes the code  
907 list schema modules that are imported into this schema module. When the  
908 `ccts:UnqualifiedDatatypes` schema module is referenced, the “`udt`” namespace prefix  
909 must be used.

910 [NMS17] The `ccts:UnqualifiedDatatypes` schema module namespace MUST be  
911 represented by the token “`udt`” when referenced in other schemas.

912

#### 913 *3.7.4.9.2 UBL Qualified Datatypes Schema Module*

914 The `ubl:QualifiedDatatype` is defined by specifying restrictions on the  
915 `ccts:UnqualifiedDatatype`. To align the UBL qualified Datatypes  
916 (`ubl:QualifiedDatatypes`) with the UBL modularity and reuse goals, the creation of a single  
917 schema module that defines all `ubl:QualifiedDatatypes` is required.

918 [SSM18] A schema module defining all UBL Qualified Datatypes MUST be created.

919 The `ubl:QualifiedDatatypes` must be based upon the `ccts:UnqualifiedDatatypes`.

920 [SSM20] The UBL Qualified Datatypes schema module MUST import the  
921 `ccts:UnQualifiedDatatypes` schema module.

922 The `ubl:QualifiedDatatypes` schema module name must follow the UBL module naming  
923 approach.

924 [SSM19] The UBL Qualified Datatypes schema module MUST be identified as  
925 `QualifiedDatatypes` in the document name in the schema header.

#### 926 *3.7.4.9.3 UBL Qualified Datatypes Schema Module Namespace*

927 In keeping with the overall UBL namespace approach, a singular namespace must be created for  
928 storing the `ubl:QualifiedDatatypes` schema module.

929 [NMS15] The `ubl:QualifiedDatatypes` schema module MUST reside in its own  
930 namespace.

931 To ensure consistency in expressing the `ubl:QualifiedDatatypes` schema module, a  
932 normative token that will be used in all UBL schemas must be defined.

933 [NMS16] The `ubl:QualifiedDatatypes` schema module namespace MUST be  
934 represented by the namespace prefix “`qdt`” when referenced in other schemas.

935 To ensure consistency in expressing the `CommonExtensionComponents` schema module, a  
936 normative token that will be used in all UBL schemas must be defined.

937 [NMS18] The `CommonExtensionComponents` schema module namespace MUST be  
938 represented by the namespace prefix 'ext' when referenced in other schemas.

## 939 3.8 Annotation and Documentation Requirements

940 Annotation is an essential tool in understanding and reusing a schema. UBL, as an  
941 implementation of CCTS, requires an extensive amount of annotation to provide all necessary  
942 metadata required by the CCTS specification. Each construct declared or defined within the UBL  
943 library contains the requisite associated metadata to fully describe its nature and support the  
944 CCTS requirement.

### 945 3.8.1 Schema Annotation

946 Although the UBL schema annotation is necessary, its volume results in a considerable increase  
947 in the size of the UBL schemas with undesirable performance impacts. To address this issue, two  
948 schemas will be developed for each UBL schema. A normative, fully annotated schema will be  
949 provided to facilitate greater understanding of the schema module and its components, and to  
950 meet the CCTS metadata requirements. A non-normative schema devoid of annotation will also  
951 be provided that can be used at run-time if required to meet processor resource constraints.

952 [GXS2] UBL MUST provide two schemas for each transaction. One normative schema shall  
953 be fully annotated. One non-normative schema shall be a run-time schema devoid of  
954 documentation.

### 955 3.8.2 Embedded documentation

956 The information about each UBL `ccts:BusinessInformationEntity` is in the UBL  
957 spreadsheet models. UBL spreadsheets contain all necessary information to produce fully  
958 annotated schemas. Fully annotated schemas are valuable tools to implementers to assist in  
959 understanding the nuances of the information contained therein. UBL annotations will consist of  
960 information currently required by Section 7 of the CCTS and supplemented by metadata from the  
961 UBL spreadsheet models.

962 The absence of an optional annotation inside the structured set of annotations in the  
963 documentation element implies the use of the default value. For example, there are several  
964 annotations relating to context such as `ccts:BusinessContext` or `ccts:IndustryContext`  
965 whose absence implies that their value is "all contexts".

966 The following rules describe the documentation requirements for each  
967 `ubl:QualifiedDatatype` and `ccts:UnqualifiedDatatype` definition. None of these  
968 documentation rules apply in the case of extension where the 'UBL Extensions' element is used.

969 [DOC1] The `xsd:documentation` element for every `Datatype` MUST contain a structured set of  
970 annotations in the following sequence and pattern (as defined in CCTS Section 7):

- 971 • `DictionaryEntryName` (mandatory)
- 972 • `Version` (mandatory):
- 973 • `Definition`(mandatory)



- 974 • RepresentationTerm (mandatory)
- 975 • QualifierTerm(s) (mandatory, where used)
- 976 • UniqueIdentifier (mandatory)
- 977 • Usage Rule(s) (optional)
- 978 • Content Component Restriction (optional)

979  
 980 [DOC2] A Datatype definition MAY contain one or more Content Component Restrictions to  
 981 provide additional information on the relationship between the Datatype and its  
 982 corresponding Core Component Type. If used the Content Component Restrictions  
 983 must contain a structured set of annotations in the following patterns:

- 984 • RestrictionType (mandatory): Defines the type of format restriction that applies to the  
 985 Content Component.
- 986 • RestrictionValue (mandatory): The actual value of the format restriction that applies  
 987 to the Content Component.
- 988 • ExpressionType (optional): Defines the type of the regular expression of the  
 989 restriction value.

990  
 991 [DOC3] A Datatype definition MAY contain one or more Supplementary Component Restrictions  
 992 to provide additional information on the relationship between the Datatype and its  
 993 corresponding Core Component Type. If used the Supplementary Component  
 994 Restrictions must contain a structured set of annotations in the following patterns:

- 995 • SupplementaryComponentName (mandatory): Identifies the Supplementary  
 996 Component on which the restriction applies.
- 997 • RestrictionValue (mandatory, repetitive): The actual value(s) that is (are) valid  
 998 for the Supplementary Component

999 The following rule describes the documentation requirements for each `ccts:Basic`  
 1000 `BusinessInformationEntity` definition.

1001 [DOC4] The `xsd:documentation` element for every Basic Business Information Entity MUST  
 1002 contain a structured set of annotations in the following patterns:

- 1003 • ComponentType (mandatory): The type of component to which the object belongs.  
 1004 For Basic Business Information Entities this must be "BBIE".
- 1005 • DictionaryEntryName (mandatory): The official name of a Basic Business Information  
 1006 Entity.
- 1007 • Version (optional): An indication of the evolution over time of the Basic Business  
 1008 Information Entity.
- 1009 • Definition(mandatory): The semantic meaning of a Basic Business Information Entity.
- 1010 • Cardinality(mandatory): Indication whether the Basic Business Information Entity  
 1011 represents a not-applicable, optional, mandatory and/or repetitive characteristic of the  
 1012 Aggregate Business Information Entity.
- 1013 • ObjectClassQualifier (optional): The qualifier for the object class.
- 1014 • ObjectClass(mandatory): The Object Class containing the Basic Business  
 1015 Information Entity.

- 1016 • PropertyTermQualifier (optional): A qualifier is a word or words which help define and  
1017 differentiate a Basic Business Information Entity.
- 1018 • PropertyTerm(mandatory): Property Term represents the distinguishing characteristic  
1019 or Property of the Object Class and shall occur naturally in the definition of the Basic  
1020 Business Information Entity.
- 1021 • RepresentationTerm (mandatory): A Representation Term describes the form in  
1022 which the Basic Business Information Entity is represented.
- 1023 • DataTypeQualifier (optional): semantically meaningful name that differentiates the  
1024 Datatype of the Basic Business Information Entity from its underlying Core  
1025 Component Type.
- 1026 • DataType (mandatory): Defines the Datatype used for the Basic Business Information  
1027 Entity.
- 1028 • AlternativeBusinessTerms (optional): Any synonym terms under which the Basic  
1029 Business Information Entity is commonly known and used in the business.
- 1030 • Examples (optional): Examples of possible values for the Basic Business Information  
1031 Entity.

1032 The following rule describes the documentation requirements for each  
1033 `ccts:AggregateBusinessInformationEntity` definition.

- 1034 [DOC5] The `xsd:documentation` element for every Aggregate Business Information Entity  
1035 MUST contain a structured set of annotations in the following sequence and pattern:
- 1036 • ComponentType (mandatory): The type of component to which the object belongs.  
1037 For Aggregate Business Information Entities this must be "ABIE".
  - 1038 • DictionaryEntryName (mandatory): The official name of the Aggregate Business  
1039 Information Entity .
  - 1040 • Version (optional): An indication of the evolution over time of the Aggregate Business  
1041 Information Entity.
  - 1042 • Definition(mandatory): The semantic meaning of the Aggregate Business Information  
1043 Entity.
  - 1044 • ObjectClassQualifier (optional): The qualifier for the object class.
  - 1045 • ObjectClass(mandatory): The Object Class represented by the Aggregate Business  
1046 Information Entity.
  - 1047 • AlternativeBusinessTerms (optional): Any synonym terms under which the Aggregate  
1048 Business Information Entity is commonly known and used in the business.

1049 The following rule describes the documentation requirements for each  
1050 `ccts:AssociationBusinessInformationEntity` definition.

- 1051 [DOC6] The `xsd:documentation` element for every Association Business Information Entity  
1052 element declaration MUST contain a structured set of annotations in the following  
1053 sequence and pattern:
- 1054 • ComponentType (mandatory): The type of component to which the object belongs.  
1055 For Association Business Information Entities this must be "ASBIE".
  - 1056 • DictionaryEntryName (mandatory): The official name of the Association Business  
1057 Information Entity.

- 1058 • Version (optional): An indication of the evolution over time of the Association  
1059 Business Information Entity.
- 1060 • Definition(mandatory): The semantic meaning of the Association Business  
1061 Information Entity.
- 1062 • Cardinality(mandatory): Indication whether the Association Business Information  
1063 Entity represents an optional, mandatory and/or repetitive association.
- 1064 • ObjectClass(mandatory): The Object Class containing the Association Business  
1065 Information Entity.
- 1066 • PropertyTermQualifier (optional): A qualifier is a word or words which help define and  
1067 differentiate the Association Business Information Entity.
- 1068 • PropertyTerm(mandatory): Property Term represents the Aggregate Business  
1069 Information Entity contained by the Association Business Information Entity.
- 1070 • AssociatedObjectClassQualifier (optional): Associated Object Class Qualifiers  
1071 describe the 'context' of the relationship with another ABIE. That is, it is the role the  
1072 contained Aggregate Business Information Entity plays within its association with the  
1073 containing Aggregate Business Information Entity.
- 1074 • AssociatedObjectClass (mandatory); Associated Object Class is the Object Class at  
1075 the other end of this association. It represents the Aggregate Business Information  
1076 Entity contained by the Association Business Information Entity.

1077

- 1078 [DOC8] The `xsd:documentation` element for every Supplementary Component attribute  
1079 declaration MUST contain a structured set of annotations in the following sequence  
1080 and pattern:
- 1081 • Name (mandatory): Name in the Registry of a Supplementary Component of a Core  
1082 Component Type.
  - 1083 • Definition (mandatory): A clear, unambiguous and complete explanation of the  
1084 meaning of a Supplementary Component and its relevance for the related Core  
1085 Component Type.
  - 1086 • Primitive type (mandatory): PrimitiveType to be used for the representation of the  
1087 value of a Supplementary Component.
  - 1088 • Possible Value(s) (optional): one possible value of a Supplementary Component.

1089

- 1090 [DOC9] The `xsd:documentation` element for every Supplementary Component attribute  
1091 declaration containing restrictions MUST include the following additional information  
1092 appended to the information required by DOC8:
- 1093 • Restriction Value(s) (mandatory): The actual value(s) that is (are) valid for the  
1094 Supplementary Component.

---

## 1095 4 Naming Rules

1096 The rules in this section make use of the following special concepts related to XML elements.

1097 Top-level element: An element that encloses a whole UBL business message. Note that UBL  
1098 business messages might be carried by messaging transport protocols that themselves have  
1099 higher-level XML structure. Thus, a UBL top-level element is not necessarily the root element of  
1100 the XML document that carries it.

1101 Lower-level element: An element that appears inside a UBL business message. Lower-level  
1102 elements consist of intermediate and leaf level.

1103 Intermediate element: An element not at the top level that is of a complex type, only containing  
1104 other elements and possibly attributes.

1105 Leaf element: An element containing only character data (though it may also have attributes).  
1106 Note that, because of the XSD mechanisms involved, a leaf element that has attributes must be  
1107 declared as having a complex type, but a leaf element with no attributes may be declared with  
1108 either a simple type or a complex type.

### 1109 4.1 General Naming Rules

1110 In keeping with CCTS, UBL will use English as its normative language. If the UBL Library is  
1111 translated into other languages for localization purposes, these additional languages might  
1112 require additional restrictions. Such restrictions are expected to be formulated as additional rules  
1113 and published as appropriate.

1114 [GNR1] UBL XML element and type names MUST be in the English language, using the  
1115 primary English spellings provided in the Oxford English Dictionary.

1116 The CCTS adheres to the International Organization for Standardization (ISO)/International  
1117 Electrotechnical Commission (IEC) Technical Specification 11179 Information technology –  
1118 Specification and standardization of data elements. The UBL component library, as a syntax-  
1119 neutral representation, is also fully conformant to those rules. The UBL syntax-specific XSD  
1120 instantiation of the UBL component library—in some cases—refines the CCTS naming rules to  
1121 leverage the capabilities of XML and XSD. Specifically, truncation rules are applied to allow for  
1122 reuse of element names across parent element environments and to maintain brevity and clarity.  
1123 CCTS, as an implementation of 11179, furthers its basic tenets of data standardization into  
1124 higher-level constructs as expressed by the `ccts:DictionaryEntryNames` of those constructs  
1125 – such as those for `ccts:BasicBusinessInformationEntities` and  
1126 `ccts:AggregateBusinessInformationEntities`. Since UBL is an implementation of  
1127 CCTS, UBL uses CCTS dictionary entry names as the basis for UBL XML schema construct  
1128 names. UBL converts these `ccts:DictionaryEntryNames` into UBL XML schema construct  
1129 names using strict transformation rules.

1130 [GNR2] UBL XML element and type names MUST be consistently derived from CCTS  
1131 conformant dictionary entry names.

1132 Dictionary entry names contain periods, spaces, other separators, and characters not allowed by  
1133 W3C XML. These separators and characters are not appropriate for UBL XML component  
1134 names.

1135 [GNR3] UBL XML element and type names constructed from  
1136 `ccts:DictionaryEntryNames` MUST NOT include periods, spaces, other  
1137 separators, or characters not allowed by W3C XML 1.0 for XML names.

1138 Acronyms and abbreviations impact on semantic interoperability, and as such are to be avoided  
1139 to the maximum extent practicable. Since some abbreviations will inevitably be necessary, UBL  
1140 will maintain a normative list of authorized acronyms and abbreviations. The intent of this  
1141 restriction is to facilitate the use of common semantics and greater understanding.

1142 [GNR4] UBL XML element, and simple and complex type names MUST NOT use acronyms,  
1143 abbreviations, or other word truncations, except those in the list of exceptions  
1144 maintained and published by the UBL TC.

1145 UBL does not desire a proliferation of acronyms and abbreviations. An exception list will be  
1146 maintained and tightly controlled by UBL. Any additions will only occur after careful scrutiny to  
1147 include assurance that any addition is critically necessary, and that any addition will not in any  
1148 way create semantic ambiguity.

1149 Once an acronym or abbreviation has been approved, it is essential to ensuring semantic clarity  
1150 and interoperability that the acronym or abbreviation is ***always*** used.

1151 [GNR6] The acronyms and abbreviations listed in the UBL-approved list MUST always be  
1152 used in place of the word or phrase they represent.

1153 Generally speaking, the names for UBL XML constructs must always be singular. The only  
1154 exception permissible is where the concept itself is pluralized.

1155 [GNR7] UBL XML element, and type names MUST be in singular form unless the concept  
1156 itself is plural.

1157 Example:

1158 `Terms`

1159 Approved acronyms and abbreviations must be used consistently across documents. To facilitate  
1160 consistency the following rules must be applied.

1161 [GNR10] Acronyms and abbreviations at the beginning of an attribute name MUST appear in  
1162 all lower case. All other acronym and abbreviation usage in an attribute declaration  
1163 MUST appear in upper case.

1164

1165 [GNR11] Acronyms and abbreviations MUST appear in all upper case for all element  
1166 declarations and type definitions.

1167 XML is case sensitive. Consistency in the use of case for a specific XML component (element,  
1168 type) is essential to ensure every occurrence of a component is treated as the same. This is  
1169 especially true in a business-based data-centric environment such as what is being addressed by  
1170 UBL. Additionally, the use of visualization mechanisms such as capitalization techniques assist in  
1171 ease of readability and ensure consistency in application and semantic clarity. The ebXML  
1172 architecture document specifies a standard use of upper and lower camel case for expressing

1173 XML elements and attributes respectively. UBL will adhere to the ebXML standard. Specifically,  
1174 UBL element and type names will be in UpperCamelCase (UCC).

1175 [GNR8] The UpperCamelCase (UCC) convention MUST be used for naming elements and types

1176 Example:

1177 CurrencyBaseRate  
1178 CityNameType

## 1179 4.2 Type Naming Rules

1180 UBL identifies several categories of naming rules for types, namely for complex types based on  
1181 Aggregate Business Information Entities, Basic Business Information Entities, and Basic  
1182 Business Information Entity Properties.

1183 Each of these CCTS constructs have a `ccts:DictionaryEntryName` that is a fully qualified  
1184 construct based on ISO 11179. As such, these names convey explicit semantic clarity with  
1185 respect to the data being described. Accordingly, these `ccts:Dictionary`  
1186 `EntryNames` provide a mechanism for ensuring that UBL `xsd:complexType` names are  
1187 semantically unambiguous, and that there are no duplications of UBL type names.

### 1188 4.2.1 Complex Type Names for CCTS Aggregate Business 1189 Information Entities (ABIEs)

1190 UBL `xsd:complexType` names for ABIEs will be derived from their dictionary entry name by  
1191 removing separators to follow general naming rules, and appending the suffix “Type” to replace  
1192 the word “Details.”

1193 [CTN1] A UBL `xsd:complexType` name based on an `ccts:Aggregate`  
1194 `BusinessInformationEntity` MUST be the `ccts:Dictionary`  
1195 `EntryName` with the separators removed and with the “Details” suffix replaced  
1196 with “Type”.

1197 Example:

<code>ccts:AggregateBusiness InformationEntity</code>	<code>UBL xsd:complexType</code>
<code>Address. Details</code>	<code>AddressType</code>
<code>Financial Account. Details</code>	<code>FinancialAccountType</code>

### 1198 4.2.2 Complex Type Names for CCTS Basic Business 1199 Information Entity (BBIE) Properties

1200 All BBIE Properties are reusable across multiple BBIEs. The CCTS does not specify, but implies,  
1201 that BBIE Property names are the reusable property term and representation term of the family of  
1202 BBIEs that are based on them. The UBL `xsd:complexType` names for BBIE Properties will be  
1203 derived from the shared property and representation terms portion of the dictionary entry names  
1204 in which they appear by removing separators to follow general naming rules, and appending the  
1205 suffix “Type”.

1206 [CTN2] A UBL `xsd:complexType` name based on a `ccts:BasicBusiness`  
1207 `InformationEntityProperty` MUST be the `ccts:Dictionary`  
1208 `EntryName` shared property term and its qualifiers and representation term of the  
1209 `ccts:BasicBusinessInformationEntity`, with the separators removed and  
1210 with the "Type" suffix appended after the representation term.

1211 **Example:**

```
1212 <!--==== Basic Business Information Entity Type Definitions ==== -->  
1213 <xsd:complexType name="ChargeIndicatorType">  
1214     ...  
1215 </xsd:complexType>
```

1216

1217 [CTN6] A UBL `xsd:complexType` name based on a `ccts:BasicBusiness`  
1218 `InformationEntityProperty` and with a `ccts:BasicBusiness`  
1219 `InformationEntityRepresentationTerm` of 'Text' MUST have the word  
1220 "Text" removed from the end of its name.

1221

1222 [CTN7] A UBL `xsd:complexType` name based on a `ccts:BasicBusiness`  
1223 `InformationEntityProperty` and with a `ccts:BasicBusiness`  
1224 `InformationEntityRepresentationTerm` of 'Identifier' MUST have  
1225 the word "Identifier" replaced by the word "ID" at the end of its name.

1226

1227 [CTN8] A UBL `xsd:complexType` name based on a `ccts:BasicBusiness`  
1228 `InformationEntityProperty` MUST remove all duplication of words that occur  
1229 as a result of duplicate property terms and representation terms.

## 1230 4.3 Element Naming Rules

1231 As defined in the UBL Model (See Figure 2-3), UBL elements will be created for  
1232 `ccts:AggregateBusinessInformationEntities`, `ccts:BasicBusiness`  
1233 `InformationEntities`, and `ccts:AssociationBusinessInformation`  
1234 `Entities`. UBL element names will reflect this relationship in full conformance with ISO11179  
1235 element naming rules.

### 1236 4.3.1 Element Names for CCTS Aggregate Business 1237 Information Entities (ABIEs)

1238 [ELN1] A UBL global element name based on a `ccts:ABIE` MUST be the same as the  
1239 name of the corresponding `xsd:complexType` to which it is bound, with the word  
1240 "Type" removed.

1241 For example, a UBL `xsd:complexType` name based on the ABIE `Party`. Details will be  
1242 `PartyType`. The global element based on `PartyType` will be named `Party`.

#### 1243 Example:

```
1244 <xsd:element name="Party" type="PartyType"/>  
1245 <xsd:complexType name="PartyType">  
1246  
1247 <xsd:annotation>  
1248 -!--Documentation goes here--> </xsd:annotation>  
1249 <xsd:sequence>  
1250 <xsd:element ref="cbc:MarkCareIndicator" minOccurs="0" maxOccurs="1">  
1251 ...  
1252 </xsd:element>  
1253 <xsd:element ref="cbc:MarkAttentionIndicator" minOccurs="0" maxOccurs="1">  
1254 ...  
1255 </xsd:element>  
1256 <xsd:element ref="PartyIdentification" minOccurs="0" maxOccurs="unbounded">  
1257 ...  
1258 </xsd:element>  
1259 <xsd:element ref="PartyName" minOccurs="0" maxOccurs="1">  
1260 ...  
1261 </xsd:element>  
1262 <xsd:element ref="Address" minOccurs="0" maxOccurs="1">  
1263 ...  
1264 </xsd:element>  
1265 ...  
1266 </xsd:sequence>  
1267
```

### 1268 4.3.2 Element Names for CCTS Basic Business Information 1269 Entity (BBIE) Properties

1270 The same naming concept used for ABIEs applies to BBIE Properties.

1271 [ELN2] A UBL global element name based on a `ccts:BBIEProperty` MUST be the same  
1272 as the name of the corresponding `xsd:complexType` to which it is bound, with the  
1273 word "Type" removed.

#### 1274 Example:



1275  
1276  
1277  
1278  
1279  
1280  
1281

```
<!--==== Basic Business Information Entity Type Definitions =====>  
<xsd:complexType name="ChargeIndicatorType">  
  ...  
</xsd:complexType>  
  ...  
<!--==== Basic Business Information Entity Property Element Declarations =====>  
<xsd:element name="ChargeIndicator" type="ChargeIndicatorType"/>
```

### 1282 4.3.3 Element Names for CCTS Association Business 1283 Information Entities (ASBIEs)

1284 An ASBIE is not a class like an ABIE or a BBIE Property that is reused as a BBIE. Rather, it is an  
1285 association between two classes. As such, an element representing the ASBIE does not have its  
1286 own unique `xsd:complexType`. Instead, when an element representing an ASBIE is declared,  
1287 the element is bound to the `xsd:complexType` of its associated ABIE by referencing its global  
1288 element declaration.

1289 [ELN3] A UBL global element name based on a `ccts:ASBIE` MUST be the `ccts:ASBIE`  
1290 dictionary entry name property term and its qualifiers; and the object class term and  
1291 qualifiers of its associated `ccts:ABIE`. All `ccts:DictionaryEntryName` separators MUST  
1292 be removed..

## 1293 4.4 Attributes in UBL

1294 UBL, as a transactional based XML exchange format, has chosen to significantly restrict the use  
1295 of attributes. This restriction is in keeping with the fact that attribute usage is relegated to  
1296 supplementary components only; all "primary" business data appears exclusively in element  
1297 content. These attributes are defined in the UN/CEFACT Unqualified Datatype schema module.

---

## 1298 5 Declarations and Definitions

1299 In W3C XML Schema, elements are defined in terms of complex or simple types and attributes  
1300 are defined in terms of simple types. The rules in this section govern the consistent structuring of  
1301 these type constructs and the manner for unambiguously and thoroughly documenting them in  
1302 the UBL Library.

### 1303 5.1 Type Definitions

#### 1304 5.1.1 General Type Definitions

1305 Since UBL elements and types are intended to be reusable, all types must be named. This  
1306 permits other types to establish elements that reference these types, and also supports the use of  
1307 extensions for the purposes of versioning and customization.

1308 [GTD1] All types **MUST** be named.

#### 1309 **Example:**

```
1310 <xsd:complexType name="QuantityType">  
1311   ...  
1312 </xsd:complexType>
```

1313 UBL disallows the use of the type `xsd:anyType`, because this feature permits the introduction of  
1314 potentially unknown types into an XML instance. UBL intends that all constructs within the  
1315 instance be described by the schemas describing that instance - `xsd:anyType` is seen as  
1316 working counter to the requirements of interoperability. In consequence, particular attention is  
1317 given to the need to enable meaningful validation of the UBL document instances. Were it not for  
1318 this, `xsd:anyType` might have been allowed.

1319 [GTD2] The predefined XML Schema type `xsd:anyType` **MUST NOT** be used.

#### 1320 5.1.2 Simple Types

1321 The Core Components Technical Specification provides a set of constructs for the modeling of  
1322 basic data, Core Component Types. These are represented in UBL with a library of complex  
1323 types, with the effect that most "simple" data is represented as property sets defined according to  
1324 the CCTs, made up of content components and supplementary components. In most cases, the  
1325 supplementary components are expressed as XML attributes, the content component becomes  
1326 element content, and the CCT is represented with an `xsd:complexType`. There are exceptions  
1327 to this rule in those cases where all of a CCT's properties can be expressed without the use of  
1328 attributes. In these cases, an `xsd:simpleType` is used.

1329 UBL does not define its own simple types. These are defined in the UN/CEFACT Unqualified  
1330 Datatype schema module. UBL may define restrictions of these simple types in the UBL Qualified  
1331 Datatype schema module.

### 1332 5.1.3 Complex Types

1333 Since even simple datatypes are modeled as property sets in most cases, the XML expression of  
1334 these models primarily employs `xsd:complexType`. To facilitate reuse, versioning, and  
1335 customization, all complex types are named. In the UBL model ABIEs, are considered classes  
1336 (objects) .

1337 [CTD1] For every class identified in the UBL model, a named `xsd:complexType` MUST be  
1338 defined.

#### 1339 Example:

```
1340 <xsd:complexType name="BuildingNameType">  
1341 </xsd:complexType>
```

1342 Every class identified in the UBL model consists of properties. These properties are either  
1343 ASBIEs, when the property represents another class, or BBIE properties.

1344 [CTD25] For every `ccts:BBIEProperty` identified in the UBL model a named  
1345 `xsd:complexType` must be defined.

1346

### 1347 5.1.3.10 Aggregate Business Information Entities (ABIEs)

1348 The concept of an ABIE encapsulates the relationship between a class (the ABIE) and its  
1349 properties (those data items contained within the ABIE). UBL represents this relationship by  
1350 defining an `xsd:complexType` for each ABIE with its properties represented as a sequence of  
1351 references to global elements.

1352 [CTD2] Every `ccts:ABIE` `xsd:complexType` definition content model MUST use the  
1353 `xsd:sequence` element containing references to the appropriate global element  
1354 declarations.

#### 1355 Example:

```
1356 <xsd:complexType name="AddressType">  
1357 ...  
1358 <xsd:sequence>  
1359 <xsd:element ref="cbc:CityName" minOccurs="0" maxOccurs="1">  
1360 ...  
1361 </xsd:element>  
1362 <xsd:element ref="cbc:PostalZone" minOccurs="0" maxOccurs="1">  
1363 ...  
1364 </xsd:element>...  
1365 </xsd:sequence>  
1366 </xsd:complexType>
```

### 1367 5.1.3.11 Basic Business Information Entities (BBIEs)

1368 All BBIEs, in accordance with the Core Components Technical Specification, have a  
1369 representation term. This may be a primary or secondary representation term. Representation  
1370 terms describe the structural representation of the BBIE. These representation terms are  
1371 expressed in the UBL Model as Unqualified Datatypes bound to a Core Component Type that  
1372 describes their structure. In addition to the Unqualified Datatypes defined in CCTS, UBL has

1373 defined a set of Qualified Datatypes that are derived from the CCTS Unqualified Datatypes. There  
1374 are a set of rules concerning the way these relationships are expressed in the UBL XML library.  
1375 As discussed above, BBIE Properties are represented with complex types. Within these are  
1376 `xsd:simpleContent` elements that extend the Datatypes.

1377 [CTD3] Every `ccts:BBIEProperty` `xsd:complexType` definition content model MUST  
1378 use the `xsd:simpleContent` element.  
1379

1380 [CTD4] Every `ccts:BBIEProperty` `xsd:complexType` content model  
1381 `xsd:simpleContent` element MUST consist of an `xsd:extension` element.  
1382

1383 [CTD5] Every `ccts:BBIEProperty` `xsd:complexType` content model `xsd:base`  
1384 attribute value MUST be the UN/CEFACT Unqualified Datatype or UBL Qualified  
1385 Datatype as appropriate.

1386 **Example:**

```
1387 <xsd:complexType name="StreetNameType">  
1388   <xsd:simpleContent>  
1389     <xsd:extension base="udt:NameType" />  
1390   </xsd:simpleContent>  
1391 </xsd:complexType>
```

## 1392 5.1.3.12 Datatypes

1393 There is a direct one-to-one relationship between `ccts:CoreComponentTypes` and  
1394 `ccts:PrimaryRepresentationTerms`. Additionally, there are several  
1395 `ccts:SecondaryRepresentationTerms` that are semantic refinements of their parent  
1396 `ccts:PrimaryRepresentationTerm`. The total set of `ccts:RepresentationTerms` by  
1397 their nature represent `ccts:Datatypes`. Specifically, for each  
1398 `ccts:PrimaryRepresentationTerm` or `ccts:SecondaryRepresentationTerm`, a  
1399 `ccts:UnqualifiedDatatype` exists. In the UBL XML Library, these  
1400 `ccts:UnqualifiedDatatypes` are expressed as complex or simple types that are of the type  
1401 of its corresponding `ccts:CoreComponentType`. UBL uses the  
1402 `ccts:UnqualifiedDatatypes` that are provided by the UN/CEFACT Unqualified Datatype  
1403 (`udt`) schema module.

### 1404 5.1.3.12.1 Qualified Datatypes

1405 The data types defined in the unqualified data type schema module are intended to be suitable as  
1406 the `xsd:base` type for some, but not all BBIEs. As business process modeling reveals the need  
1407 for specialized data types, new 'qualified' types will need to be defined. These new  
1408 `ccts:QualifiedDatatype` must be based on an `ccts:UnqualifiedDatatype` and must  
1409 represent a semantic or technical restriction of the `ccts:UnqualifiedDatatype`. Technical  
1410 restrictions must be implemented as a `xsd:restriction` or as a new `xsd:simpleType` if the  
1411 supplementary components of the qualified data type map directly to the properties of a built-in  
1412 XSD data type.

1413 [CTD6] For every Qualified Datatype used in the UBL model, a named `xsd:complexType`  
1414 or `xsd:simpleType` MUST be defined.

1415

1416 [CTD20] A `ccts:QualifiedDataType` MUST be based on an unqualified data type and  
1417 add some semantic and/or technical restriction to the unqualified data type.

1418

1419 [CTD21] The name of a `ccts:QualifiedDataType` MUST be the name of its base  
1420 `ccts:UnqualifiedDataType` with separators and spaces removed and with its  
1421 qualifier term added.

1422 In accordance with rule GXS3 built-in XSD data types will be used whenever possible.

1423 [CTD22] Every qualified datatype based on an unqualified datatype `xsd:complexType`  
1424 whose supplementary components map directly to the properties of an XSD built-in  
1425 data type  
1426 MUST be defined as an `xsd:simpleType`  
1427 MUST contain one `xsd:restriction` element  
1428 MUST include an `xsd:base` attribute that defines the specific XSD built-in data type  
1429 required for the content component

1430

1431 [CTD23] Every qualified datatype based on an unqualified datatype `xsd:complexType`  
1432 whose supplementary components do not map directly to the properties of an XSD  
1433 built-in data type  
1434 MUST be defined as an `xsd:complexType`  
1435 MUST contain one `xsd:simpleContent` element  
1436 MUST contain one `xsd:restriction` element  
1437 MUST include the unqualified datatype as its `xsd:base` attribute

1438

1439 [CTD24] Every qualified datatype based on an unqualified datatype `xsd:simpleType`  
1440 MUST contain one `xsd:restriction` element  
1441 MUST include the unqualified datatype as its `xsd:base` attribute

### 1442 5.1.3.13 Core Component Types

1443 UBL has adopted UN/CEFACT's Core Component Type schema module.

## 1444 5.2 Element Declarations

### 1445 5.2.1 Elements Bound to Complex Types

1446 The binding of UBL elements to their `xsd:complexType` is based on the associations identified  
1447 in the UBL model. For the `ccts:BasicBusinessInformationEntities` (BBIEs) and  
1448 `ccts:AggregateBusinessInformationEntities` (ABIEs), the UBL elements will be  
1449 directly associated to its corresponding `xsd:complexType`.

1450 [ELD3] For every class and property identified in the UBL model, a global element bound to  
1451 the corresponding `xsd:complexType` MUST be declared.

1452 **Example:**

1453 For the `Party.Details` object class, a complex type/global element declaration pair is  
1454 created through the declaration of a `Party` element that is of type `PartyType`.

1455 The element thus created is useful for reuse in the building of new business messages. The  
1456 complex type thus created is useful for both reuse and customization, in the building of both new  
1457 and contextualized business messages.

1458 **Example:**

```
1459 <xsd:element name="BuyerParty" type="BuyerPartyType"/>  
1460 <xsd:complexType name="BuyerPartyType" ...  
1461 </xsd:complexType>
```

## 1462 5.2.2 Elements Representing ASBIEs

1463 A `ccts:AssociationBusinessInformationEntity` (ASBIE) is not a class like ABIEs.  
1464 Rather, it is an association between two classes. As such, the element declaration will bind the  
1465 element to the `xsd:complexType` of the associated ABIE. There are two types of ASBIEs – those  
1466 that have qualifiers in the object class, and those that do not.

1467 [ELD4] When a `ccts:ASBIE` is unqualified, it is bound via reference to the global  
1468 `ccts:ABIE` element to which it is associated.

1469

1470 [ELD11] When a `ccts:ASBIE` is qualified, a new element MUST be declared and bound to  
1471 the `xsd:complexType` of its associated `ccts:ABIE`.

## 1472 5.2.3 Code List Import

1473 [ELD6] The code list `xsd:import` element MUST contain the namespace and schema  
1474 location attributes.

## 1475 5.2.4 Empty Elements

1476 [ELD7] Empty elements MUST not be declared, except in the case of extension, where the  
1477 'UBL Extensions' element is used.

---

1478 **6 Code Lists**

1479 UBL has adopted the Code List Methodology proposed by G Ken Holman. See the UBL TC site  
1480 for a link to the latest draft.

1481 In addition to the methodology, the following rules apply.

1482 [CDL1] All UBL Codes **MUST** be part of a UBL or externally maintained Code List.

1483 Because the majority of code lists are owned and maintained by external agencies, UBL will  
1484 make maximum use of such external code lists where they exist.

1485 [CDL2] The UBL Library **SHOULD** identify and use external standardized code lists rather  
1486 than develop its own UBL-native code lists.

1487 In some cases the UBL Library may extend an existing code list to meet specific business  
1488 requirements. In others cases the UBL Library may have to create and maintain a code list where  
1489 a suitable code list does not exist in the public domain. Both of these types of code lists would be  
1490 considered UBL-internal code lists.

1491 [CDL3] The UBL Library **MAY** design and use an internal code list where an existing external  
1492 code list needs to be extended, or where no suitable external code list exists.

1493 UBL-internal code lists will be designed with maximum re-use in mind to facilitate maximum use  
1494 by others.

---

## 1495 7 Miscellaneous XSD Rules

1496 UBL, as a business standard vocabulary, requires consistency in its development. The number of  
1497 UBL Schema developers will expand over time. To ensure consistency, it is necessary to address  
1498 the optional features in XSD that are not addressed elsewhere.

### 1499 7.1 xsd:simpleType

1500 UBL guiding principles require maximum reuse. XSD provides for forty four built-in Datatypes  
1501 expressed as simple types. In keeping with the maximize re-use guiding principle, these built-in  
1502 simple types should be used wherever possible.

1503 [GXS3] Built-in XSD Simple Types SHOULD be used wherever possible.

### 1504 7.2 Namespace Declaration

1505 The W3C XSD specification allows for the use of any token to represent its location. To ensure  
1506 consistency, UBL has adopted the generally accepted convention of using the “xsd” token for all  
1507 UBL schema and schema modules.

1508 [GXS4] All W3C XML Schema constructs in UBL Schema and schema modules MUST  
1509 contain the following namespace declaration on the xsd schema element:

1510 `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`

### 1511 7.3 xsd:substitutionGroup

1512 The xsd:substitutionGroup feature enables a type definition to identify substitution elements in a  
1513 group. Although a useful feature in document centric XML applications, this feature is not used by  
1514 UBL.

1515 [GXS5] The xsd:substitutionGroup feature MUST NOT be used.

### 1516 7.4 xsd:final

1517 UBL does not use extensions in its normative schema. Extensions are allowed by customizers as  
1518 outlined in the Guidelines for Customization. UBL may determine that certain type definitions are  
1519 inappropriate for any customization. In those instances, the xsd:final attribute will be used.

1520 [GXS6] The xsd:final attribute MUST be used to control extensions where there is a  
1521 desire to prohibit further extensions.

### 1522 7.5 xsd: notation

1523 The xsd:notation attribute identifies a notation. Notation declarations corresponding to all the  
1524 <notation> element information items in the [children], if any, plus any included or  
1525 imported declarations. Per XSD Part 2, “It is an -error- for NOTATION to be used directly in a  
1526 schema. Only Datatypes that are -derived- from NOTATION by specifying a value for



1527 -enumeration- can be used in a schema.” The UBL schema model does not require or support the  
1528 use of this feature.

1529 [GXS7] `xsd:notation` MUST NOT be used.

## 1530 7.6 `xsd:all`

1531 The `xsd:all` compositor requires occurrence indicators of `minOccurs = 0` and `maxOccurs =`  
1532 `1`. The `xsd:all` compositor allows for elements to occur in any order. The result is that in an  
1533 instance document, elements can occur in any order, are always optional, and never occur more  
1534 than once. Such restrictions are inconsistent with data-centric scenarios such as UBL.

1535 [GXS8] The `xsd:all` element MUST NOT be used.

## 1536 7.7 `xsd:choice`

1537 The `xsd:choice` compositor allows for any element declared inside it to occur in the instance  
1538 document, but only one. As with the `xsd:all` compositor, this feature is inconsistent with  
1539 business transaction exchanges. UBL recognizes that it is a very useful construct in situations  
1540 where customization and extensibility are not a concern, however, UBL does not recommend its  
1541 use because `xsd:choice` cannot be extended.

1542 [GXS9] The `xsd:choice` element SHOULD NOT be used where customisation and  
1543 extensibility are a concern.

## 1544 7.8 `xsd:include`

1545 `xsd:include` can only be used when the including schema is in the same namespace as the  
1546 included schema.

## 1547 7.9 `xsd:union`

1548 The `xsd:union` feature provides a mechanism whereby a datatype is created as a union of two  
1549 or more existing datatypes. With UBL’s strict adherence to the use of `cts:Datatypes` that are  
1550 explicitly declared in the UBL library, this feature is inappropriate except for codelists. In some  
1551 cases external customizers may choose to use this technique for codelists and as such the use of  
1552 the union technique may prove beneficial for customizers.

1553 [GXS11] The `xsd:union` technique MUST NOT be used except for Code Lists. The `xsd:union`  
1554 technique MAY be used for Code Lists.

## 1555 7.10 `xsd:appinfo`

1556 The `xsd:appinfo` feature is used by schema to convey processing instructions to a processing  
1557 application, Stylesheet, or other tool. Some users of UBL have determined that this technique  
1558 poses a security risk and have employed techniques for stripping `xsd:appinfo` from schemas.  
1559 As UBL is committed to ensuring the widest possible target audience for its XML library, this  
1560 feature is not used – except to convey non-normative information.

1561 [GXS12] UBL designed schema SHOULD NOT use `xsd:appinfo`. If used, `xsd:appinfo`  
1562 MUST only be used to convey non-normative information.

## 1563 7.11 `xsd:schemaLocation`

1564 UBL is an international standard that will be used in perpetuity by companies around the globe. It  
1565 is important that these users have unfettered access to all UBL schema.

1566 [GXS15] Each `xsd:schemaLocation` attribute declaration MUST contain a system-resolvable  
1567 URL, which at the time of release from OASIS shall be a relative URL referencing the  
1568 location of the schema or schema module in the release package.

## 1569 7.12 `xsd:nillable`

1570 [GXS16] The built in `xsd:nillable` attribute MUST NOT be used for any UBL declared  
1571 element.

## 1572 7.13 `xsd:anyAttribute`

1573 UBL disallows the use of `xsd:anyAttribute`, because this feature permits the introduction of  
1574 potentially unknown attributes into an XML instance. UBL intends that all constructs within the  
1575 instance be described by the schemas describing that –instance– `xsd:anyAttribute` is seen  
1576 as working counter to the requirements of interoperability. In consequence, particular attention is  
1577 given to the need to enable meaningful validation of the UBL document instances. Were it not for  
1578 this, `xsd:anyAttribute` might have been allowed.

1579 [GXS17] The `xsd:anyAttribute` MUST NOT be used.

## 1580 7.14 Extension and Restriction

1581 UBL fully recognizes the value of supporting extension and restriction of its core library by  
1582 customizers. The UBL extension and restriction recommendations are discussed in the  
1583 *Guidelines for the Customization of UBL Schemas* available as part of UBL 1.0.

1584 [GXS13] Complex Type extension or restriction MAY be used where appropriate.

---

1585

## 8 Instance Documents

1586 Previous drafts of this document contained a section specifying several rules governing  
1587 conformant UBL instances. Since these rules, addressing instance validation, character  
1588 encoding, and empty elements, do not pertain to schema design or the naming of information  
1589 items, they have been relocated to the UBL 2.0 specification as document constraints to be  
1590 observed in addition to the constraints expressed in the UBL 2.0 schemas.

---

## 1591 **Appendix A. UBL NDR 2.0 Checklist**

1592 The following checklist constitutes all UBL XML naming and design rules as defined in *UBL*  
1593 *Naming and Design Rules version 2.0*, 26 January 2006. The checklist is in alphabetical  
1594 sequence as follows:

- 1595 Attribute Declaration Rules (ATD)
- 1596 Code List Rules (CDL)
- 1597 ComplexType Definition Rules (CTD)
- 1598 ComplexType Naming Rules (CTN)
- 1599 Documentation Rules (DOC)
- 1600 Element Declaration Rules (ELD)
- 1601 Element Naming Rules (ELN)
- 1602 General Naming Rules (GNR)
- 1603 General Type Definition Rules (GTD)
- 1604 General XML Schema Rules (GXS)
- 1605 Modeling Constraints Rules (MDC)
- 1606 Naming Constraints Rules (NMC)
- 1607 Namespace Rules (NMS)
- 1608 Root Element Declaration Rules (RED)
- 1609 Schema Structure Modularity Rules (SSM)
- 1610 Versioning Rules (VER)

<b>A.1 Attribute Declaration rules</b>	
[ATD6]	(See GXS15)
[ATD7]	(See GXS16)
[ATD8]	(See GXS17)

1611

<h2>A.2 Code List rules</h2>	
[CDL1]	All UBL Codes MUST be part of a UBL or externally maintained Code List.
[CDL2]	The UBL Library SHOULD identify and use external standardized code lists rather than develop its own UBL-native code lists.
[CDL3]	The UBL Library MAY design and use an internal code list where an existing external code list needs to be extended, or where no suitable external code list exists.

1612

<h2>A.3 ComplexType Definition rules</h2>	
[CTD1]	For every class identified in the UBL model, a named <code>xsd:complexType</code> MUST be defined.
[CTD2]	Every <code>ccts:ABIE</code> <code>xsd:complexType</code> definition content model MUST use the <code>xsd:sequence</code> element containing references to the appropriate global element declarations.
[CTD3]	Every <code>ccts:BBIEProperty</code> <code>xsd:complexType</code> definition content model MUST use the <code>xsd:simpleContent</code> element.
[CTD4]	Every <code>ccts:BBIEProperty</code> <code>xsd:complexType</code> content model <code>xsd:simpleContent</code> element MUST consist of an <code>xsd:extension</code> element.
[CTD5]	Every <code>ccts:BBIEProperty</code> <code>xsd:complexType</code> content model <code>xsd:base</code> attribute value MUST be the UN/CEFACT Unqualified Datatype or UBL qualified Datatype as appropriate.
[CTD6]	For every Qualified Datatype used in the UBL model, a named <code>xsd:complexType</code> or <code>xsd:simpleType</code> MUST be defined.
[CTD20]	A <code>ccts:QualifiedDataType</code> MUST be based on an unqualified data type and add some semantic and/or technical restriction to the unqualified data type.
[CTD21]	The name of a <code>ccts:QualifiedDataType</code> MUST be the name of its base <code>ccts:UnqualifiedDataType</code> with separators and spaces removed and with its qualifier term added.

[CTD22]	<p>Every qualified datatype based on an unqualified datatype <code>xsd:complexType</code> whose supplementary components map directly to the properties of an XSD built-in data type</p> <p>MUST be defined as an <code>xsd:simpleType</code></p> <p>MUST contain one <code>xsd:restriction</code> element</p> <p>MUST include an <code>xsd:base</code> attribute that defines the specific XSD built-in data type required for the content component</p>
[CTD23]	<p>Every qualified datatype based on an unqualified datatype <code>xsd:complexType</code> whose supplementary components do not map directly to the properties of an XSD built-in data type</p> <p>MUST be defined as an <code>xsd:complexType</code></p> <p>MUST contain one <code>xsd:simpleContent</code> element</p> <p>MUST contain one <code>xsd:restriction</code> element</p> <p>MUST include the unqualified datatype as its <code>xsd:base</code> attribute</p>
[CTD24]	<p>Every qualified datatype based on an unqualified datatype <code>xsd:simpleType</code></p> <p>MUST contain one <code>xsd:restriction</code> element</p> <p>MUST include the unqualified datatype as its <code>xsd:base</code> attribute</p>
[CTD25]	<p>For every <code>ccts:BBIEProperty</code> identified in the UBL model a named <code>xsd:complexType</code> must be defined.</p>

1613

<h2 style="color: #4F81BD;">A.4 Complex Type Naming rules</h2>	
[CTN1]	<p>A UBL <code>xsd:complexType</code> name based on an <code>ccts:Aggregate BusinessInformationEntity</code> MUST be the <code>ccts:DictionaryEntryName</code> with the separators removed and with the “Details” suffix replaced with “Type”.</p>
[CTN2]	<p>A UBL <code>xsd:complexType</code> name based on a <code>ccts:BasicBusiness InformationEntityProperty</code> MUST be the <code>ccts:Dictionary EntryName</code> shared property term and its qualifiers and representation term of the <code>ccts:BasicBusinessInformationEntity</code>, with the separators removed and with the “Type” suffix appended after the representation term.</p>

[CTN6]	A UBL <code>xsd:complexType</code> name based on a <code>ccts:BasicBusinessInformationEntityProperty</code> and with a <code>ccts:BasicBusinessInformationEntityRepresentationTerm</code> of 'Text' MUST have the word "Text" removed from the end of its name.
[CTN7]	A UBL <code>xsd:complexType</code> name based on a <code>ccts:BasicBusinessInformationEntityProperty</code> and with a <code>ccts:BasicBusinessInformationEntityRepresentationTerm</code> of 'Identifier' MUST have the word "Identifier" replaced by the word "ID" at the end of its name.
[CTN8]	A UBL <code>xsd:complexType</code> name based on a <code>ccts:BasicBusinessInformationEntityProperty</code> MUST remove all duplication of words that occur as a result of duplicate property terms and representation terms.

1614

<h2>A.5 Documentation rules</h2>	
[DOC1]	<p>The <code>xsd:documentation</code> element for every Datatype MUST contain a structured set of annotations in the following sequence and pattern (as defined in CCTS Section 7):</p> <ul style="list-style-type: none"> <li>DictionaryEntryName (mandatory)</li> <li>Version (mandatory):</li> <li>Definition(mandatory)</li> <li>RepresentationTerm (mandatory)</li> <li>QualifierTerm(s) (mandatory, where used)</li> <li>UniquelIdentifier (mandatory)</li> <li>Usage Rule(s) (optional)</li> <li>Content Component Restriction (optional)</li> </ul>
[DOC2]	<p>A Datatype definition MAY contain one or more Content Component Restrictions to provide additional information on the relationship between the Datatype and its corresponding Core Component Type. If used the Content Component Restrictions must contain a structured set of annotations in the following patterns:</p> <ul style="list-style-type: none"> <li>RestrictionType (mandatory): Defines the type of format restriction that applies to the Content Component.</li> <li>RestrictionValue (mandatory): The actual value of the format restriction that</li> </ul>

	<p>applies to the Content Component.</p> <p>ExpressionType (optional): Defines the type of the regular expression of the restriction value.</p>
[DOC3]	<p>A Datatype definition MAY contain one or more Supplementary Component Restrictions to provide additional information on the relationship between the Datatype and its corresponding Core Component Type. If used the Supplementary Component Restrictions must contain a structured set of annotations in the following patterns:</p> <p>SupplementaryComponentName (mandatory): Identifies the Supplementary Component on which the restriction applies.</p> <p>RestrictionValue (mandatory, repetitive): The actual value(s) that is (are) valid for the Supplementary Component</p>
[DOC4]	<p>The <code>xsd:documentation</code> element for every Basic Business Information Entity MUST contain a structured set of annotations in the following patterns:</p> <p>ComponentType (mandatory): The type of component to which the object belongs. For Basic Business Information Entities this must be "BBIE".</p> <p>DictionaryEntryName (mandatory): The official name of a Basic Business Information Entity.</p> <p>Version (optional): An indication of the evolution over time of the Basic Business Information Entity.</p> <p>Definition(mandatory): The semantic meaning of a Basic Business Information Entity.</p> <p>Cardinality(mandatory): Indication whether the Basic Business Information Entity represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Aggregate Business Information Entity.</p> <p>ObjectClassQualifier (optional): The qualifier for the object class.</p> <p>ObjectClass(mandatory): The Object Class containing the Basic Business Information Entity.</p> <p>PropertyTermQualifier (optional): A qualifier is a word or words which help define and differentiate a Basic Business Information Entity.</p> <p>PropertyTerm(mandatory): Property Term represents the distinguishing characteristic or Property of the Object Class and shall occur naturally in the definition of the Basic Business Information Entity.</p> <p>RepresentationTerm (mandatory): A Representation Term describes the form in which the Basic Business Information Entity is represented.</p> <p>DataTypeQualifier (optional): semantically meaningful name that differentiates the</p>



	<p>Datatype of the Basic Business Information Entity from its underlying Core Component Type.</p> <p>Data<b>Type</b> (mandatory): Defines the Datatype used for the Basic Business Information Entity.</p> <p>AlternativeBusinessTerms (optional): Any synonym terms under which the Basic Business Information Entity is commonly known and used in the business.</p> <p>Examples (optional): Examples of possible values for the Basic Business Information Entity</p>
[DOC5]	<p>The <code>xsd:documentation</code> element for every Aggregate Business Information Entity <b>MUST</b> contain a structured set of annotations in the following sequence and pattern:</p> <p>ComponentType (mandatory): The type of component to which the object belongs. For Aggregate Business Information Entities this must be "ABIE".</p> <p>DictionaryEntryName (mandatory): The official name of the Aggregate Business Information Entity .</p> <p>Version (optional): An indication of the evolution over time of the Aggregate Business Information Entity.</p> <p>Definition(mandatory): The semantic meaning of the Aggregate Business Information Entity.</p> <p>ObjectClassQualifier (optional): The qualifier for the object class.</p> <p>ObjectClass(mandatory): The Object Class represented by the Aggregate Business Information Entity.</p> <p>AlternativeBusinessTerms (optional): Any synonym terms under which the Aggregate Business Information Entity is commonly known and used in the business.</p>
[DOC6]	<p>The <code>xsd:documentation</code> element for every Association Business Information Entity element declaration <b>MUST</b> contain a structured set of annotations in the following sequence and pattern:</p> <p>ComponentType (mandatory): The type of component to which the object belongs. For Association Business Information Entities this must be "ASBIE".</p> <p>DictionaryEntryName (mandatory): The official name of the Association Business Information Entity.</p> <p>Version (optional): An indication of the evolution over time of the Association Business Information Entity.</p> <p>Definition(mandatory): The semantic meaning of the Association Business Information Entity.</p>

	<p>Cardinality(mandatory): Indication whether the Association Business Information Entity represents an optional, mandatory and/or repetitive association.</p> <p>ObjectClass(mandatory): The Object Class containing the Association Business Information Entity.</p> <p>PropertyTermQualifier (optional): A qualifier is a word or words which help define and differentiate the Association Business Information Entity.</p> <p>PropertyTerm(mandatory): Property Term represents the Aggregate Business Information Entity contained by the Association Business Information Entity.</p> <p>AssociatedObjectClassQualifier (optional): Associated Object Class Qualifiers describe the 'context' of the relationship with another ABIE. That is, it is the role the contained Aggregate Business Information Entity plays within its association with the containing Aggregate Business Information Entity.</p> <p>AssociatedObjectClass (mandatory); Associated Object Class is the Object Class at the other end of this association. It represents the Aggregate Business Information Entity contained by the Association Business Information Entity.</p>
[DOC8]	<p>The <code>xsd:documentation</code> element for every Supplementary Component attribute declaration <b>MUST</b> contain a structured set of annotations in the following sequence and pattern:</p> <p>Name (mandatory): Name in the Registry of a Supplementary Component of a Core Component Type.</p> <p>Definition (mandatory): A clear, unambiguous and complete explanation of the meaning of a Supplementary Component and its relevance for the related Core Component Type.</p> <p>Primitive type (mandatory): PrimitiveType to be used for the representation of the value of a Supplementary Component.</p> <p>Possible Value(s) (optional): one possible value of a Supplementary Component.</p>
[DOC9]	<p>The <code>xsd:documentation</code> element for every Supplementary Component attribute declaration containing restrictions <b>MUST</b> include the following additional information appended to the information required by DOC8:</p> <p>Restriction Value(s) (mandatory): The actual value(s) that is (are) valid for the Supplementary Component.</p>

1615

<h2>A.6 Element Declaration rules</h2>	
[ELD2]	All element declarations <b>MUST</b> be global

[ELD3]	For every class and property identified in the UBL model, a global element bound to the corresponding <code>xsd:complexType</code> MUST be declared.
[ELD4]	When a <code>ccts:ASBIE</code> is unqualified, it is bound via reference to the global <code>ccts:ABIE</code> element to which it is associated.
[ELD6]	The code list <code>xsd:import</code> element MUST contain the namespace and schema location attributes.
[ELD7]	Empty elements MUST not be declared, except in the case of extension, where the 'UBLExtensions' element is used.
[ELD9]	(See GXS14)
[ELD11]	When a <code>ccts:ASBIE</code> is qualified, a new element MUST be declared and bound to the <code>xsd:complexType</code> of its associated <code>ccts:ABIE</code> .
[ELD12]	The 'UBLExtensions' element MUST be declared as the first child of the document element with <code>xsd:minOccurs="0"</code> .
[ELD13]	The 'UBLProfileID' element MUST be declared immediately following the 'UBLExtensions' element with <code>xsd:minOccurs="0"</code> .
[ELD14]	The 'UBLSubsetID' element MUST be declared immediately following the 'UBLProfileID' element with <code>xsd:minOccurs="0"</code> .

1616

<h2>A.7 Element Naming rules</h2>	
[ELN1]	A UBL global element name based on a <code>ccts:ABIE</code> MUST be the same as the name of the corresponding <code>xsd:complexType</code> to which it is bound, with the word "Type" removed.
[ELN2]	A UBL global element name based on a <code>ccts:BBIEProperty</code> MUST be the same as the name of the corresponding <code>xsd:complexType</code> to which it is bound, with the word "Type" removed.
[ELN3]	A UBL global element name based on a <code>ccts:ASBIE</code> MUST be the <code>ccts:ASBIE</code> dictionary entry name property term and its qualifiers; and the object class term and qualifiers of its associated <code>ccts:ABIE</code> . All <code>ccts:DictionaryEntryName</code> separators MUST be removed..

<h2>A.8 General Naming rules</h2>	
[GNR1]	UBL XML element and type names MUST be in the English language, using the primary English spellings provided in the Oxford English Dictionary.
[GNR2]	UBL XML element and type names MUST be consistently derived from CCTS conformant dictionary entry names.
[GNR3]	UBL XML element and type names constructed from <code>ccts:DictionaryEntryNames</code> MUST NOT include periods, spaces, other separators, or characters not allowed by W3C XML 1.0 for XML names
[GNR4]	UBL XML element, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions maintained and published by the UBL TC.
[GNR6]	The acronyms and abbreviations listed in the UBL-approved list MUST always be used in place of the word or phrase they represent.
[GNR7]	UBL XML element, and type names MUST be in singular form unless the concept itself is plural.
[GNR8]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.
[GNR10]	Acronyms and abbreviations at the beginning of an attribute name MUST appear in all lower case. All other acronym and abbreviation usage in an attribute declaration MUST appear in upper case.
[GNR11]	Acronyms and abbreviations MUST appear in all upper case for all element declarations and type definitions.

1617

<h2>A.9 General Type Definition Rules</h2>	
[GTD1]	All types MUST be named.
[GTD2]	The predefined XML Schema type <code>xsd:anyType</code> MUST NOT be used.

1618

## A.10 General XML Schema Rules

[GXS1]	<p>UBL Schema MUST conform to the following physical layout as applicable:</p> <pre>&lt;!-- ===== XML Declaration===== --&gt;</pre> <pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt;</pre> <pre>&lt;!-- ===== Schema Header ===== --&gt;</pre> <p style="margin-left: 40px;">Document Name: &lt; Document name as indicated in Section 3.6 &gt;</p> <p style="margin-left: 40px;">Generated On: &lt; Date schema was generated &gt;</p> <pre>&lt;!-- ===== Copyright Notice ===== --&gt;</pre> <p style="margin-left: 40px;">“Copyright „ 2001-2004 The Organization for the Advancement of Structured Information Standards (OASIS). All rights reserved.</p> <pre>&lt;!-- ===== xsd:schema Element With Namespaces Declarations ===== --&gt;</pre> <p style="margin-left: 40px;">xsd:schema element to include version attribute and namespace declarations in the following order:</p> <ul style="list-style-type: none"> <li>xmlns:xsd</li> <li>Target namespace</li> <li>Default namespace</li> <li>CommonAggregateComponents</li> <li>CommonBasicComponents</li> <li>CoreComponentTypes</li> <li>Unspecialized Unqualified Datatypes</li> <li>Specialized Qualified Datatypes</li> <li>Identifier Schemes</li> <li>Code Lists</li> <li>Attribute Declarations – elementFormDefault=""qualified"" attributeFormDefault=""unqualified""</li> <li style="margin-left: 80px;">Version Attribute</li> </ul> <pre>&lt;!-- ===== Imports ===== --&gt;</pre>
--------	---

	<p>CommonAggregateComponents schema module</p> <p>CommonBasicComponents schema module</p> <p>Unspecialized Unqualified Types schema module</p> <p>Specialized Qualified Types schema module</p> <p>&lt;!-- ===== Global Attributes ===== --&gt;</p> <p>Global Attributes and Attribute Groups</p> <p>&lt;!-- ===== Root Element ===== --&gt;</p> <p>Root Element Declaration</p> <p>Root Element Type Definition</p> <p>&lt;!-- ===== Element Declarations ===== --&gt;</p> <p>alphabetized order</p> <p>&lt;!-- ===== Type Definitions ===== --&gt;</p> <p>All type definitions segregated by basic and aggregates as follows</p> <p>&lt;!-- ===== Aggregate Business Information Entity Type Definitions ===== --&gt;</p> <p>alphabetized order of ccts:AggregateBusinessInformationEntity xsd:TypeDefinitions</p> <p>&lt;!-- =====Basic Business Information Entity Type Definitions ===== --&gt;</p> <p>alphabetized order of ccts:BasicBusinessInformationEntities</p> <p>&lt;!-- ===== Copyright Notice ===== --&gt;</p> <p>Required OASIS full copyright notice.</p>
[GXS2]	<p>UBL MUST provide two schemas for each transaction. One normative schema shall be fully annotated. One non-normative schema shall be a run-time schema devoid of documentation..</p>
[GXS3]	<p>Built-in XSD Simple Types SHOULD be used wherever possible.</p>
[GXS4]	<p>All W3C XML Schema constructs in UBL Schema and schema modules MUST contain the following namespace declaration on the xsd schema element: xmlns:xsd="http://www.w3.org/2001/XMLSchema"</p>

[GXS5]	The <code>xsd:substitutionGroup</code> feature MUST NOT be used.
[GXS6]	The <code>xsd:final</code> attribute MUST be used to control extensions where there is a desire to prohibit further extensions.
[GXS7]	<code>xsd:notation</code> MUST NOT be used.
[GXS8]	The <code>xsd:all</code> element MUST NOT be used.
[GXS9]	The <code>xsd:choice</code> element SHOULD NOT be used where customisation and extensibility are a concern.
[GXS11]	The <code>xsd:union</code> technique MUST NOT be used except for Code Lists. The <code>xsd:union</code> technique MAY be used for Code Lists.
[GXS12]	UBL designed schema SHOULD NOT use <code>xsd:appinfo</code> . If used, <code>xsd:appinfo</code> MUST only be used to convey non-normative information.
[GXS13]	Complex Type extension or restriction MAY be used where appropriate.
[GXS14]	The <code>xsd:any</code> element MUST NOT be used except within the 'ExtensionContentType' type definition, and with <code>xsd:processContents="skip"</code> for non-UBL namespaces.
[GXS15]	Each <code>xsd:schemaLocation</code> attribute declaration MUST contain a system-resolvable URL, which at the time of release from OASIS shall be a relative URL referencing the location of the schema or schema module in the release package.
[GXS16]	The built in <code>xsd:nillable</code> attribute MUST NOT be used for any UBL declared element.
[GXS17]	The <code>xsd:anyAttribute</code> MUST NOT be used.

1619

<h2>A.11 Modelling constraint rules</h2>	
[MDC1]	UBL Libraries and Schemas MUST only use ebXML Core Component approved <code>cts:CoreComponentTypes</code> , except in the case of extension, where the 'UBL Extensions' element is used
[MDC2]	Mixed content MUST NOT be used except where contained in an

	xsd:documentation element
--	---------------------------

1620

<h2>A.12 Naming constraint rules</h2>	
[NMC1]	Each dictionary entry name MUST define one and only one fully qualified path (FQP) for an element or attribute.

1621

<h2>A.13 Namespace Rules</h2>	
[NMS1]	Every UBL-defined –or -used schema module, except internal schema modules, MUST have a namespace declared using the <code>xsd:targetNamespace</code> attribute.
[NMS2]	Every UBL-defined-or -used major version schema set MUST have its own unique namespace.
[NMS3]	UBL namespaces MUST only contain UBL developed schema modules.
[NMS4]	The namespace names for UBL Schemas holding committee draft status MUST be of the form:  <code>urn:oasis:names:tc:ubl:schema:&lt;subtype&gt;:&lt;document-id&gt;</code>
[NMS5]	The namespace names for UBL Schemas holding OASIS Standard status MUST be of the form:  <code>urn:oasis:names:specification:ubl:schema:&lt;subtype&gt;:&lt;document-id&gt;</code>
[NMS6]	UBL published namespaces MUST never be changed.
[NMS7]	The <code>ubl:CommonAggregateComponents</code> schema module MUST reside in its own namespace.
[NMS8]	The <code>ubl:CommonAggregateComponents</code> schema module namespace MUST be represented by the namespace prefix “cac” when referenced in other schemas.
[NMS9]	The <code>ubl:CommonBasicComponents</code> schema module MUST reside in its own namespace.



[NMS10]	The <code>UBL:CommonBasicComponents</code> schema module namespace MUST be represented by the namespace prefix “cbc” when referenced in other schemas.
[NMS15]	The <code>ubl:QualifiedDatatypes</code> schema module MUST reside in its own namespace.
[NMS16]	The <code>ubl:QualifiedDatatypes</code> schema module namespace MUST be represented by the namespace prefix “qdt” when referenced in other schemas.
[NMS17]	The <code>ccts:UnqualifiedDatatypes</code> schema module namespace MUST be represented by the token “udt” when referenced in other schemas.
[NMS18]	The <code>CommonExtensionComponents</code> schema module namespace MUST be represented by the namespace prefix 'ext' when referenced in other schemas.

1622

<h2>A.14 Root element declaration rules</h2>	
[RED2]	The root element MUST be the only global element declared in document schemas.

1623

<h2>A.15 Schema structure modularity rules</h2>	
[SSM1]	UBL Schema expressions MAY be split into multiple schema modules.
[SSM2]	A document schema in one UBL namespace that is dependent upon type definitions or element declarations defined in another namespace MUST only import the document schema from that namespace.
[SSM3]	A document schema in one UBL namespace that is dependant upon type definitions or element declarations defined in another namespace MUST NOT import internal schema modules from that namespace.
[SSM5]	UBL schema modules MUST either be treated as external schema modules or as internal schema modules of the document schema.
[SSM6]	All UBL internal schema modules MUST be in the same namespace as their corresponding document schema.
[SSM7]	Each UBL internal schema module MUST be named

	{ParentSchemaModuleName}{InternalSchemaModuleFunction}{schema module}
[SSM8]	A UBL schema module MAY be created for reusable components.
[SSM9]	A schema module defining all UBL Common Aggregate Components MUST be created.
[SSM10]	The UBL Common Aggregate Components schema module MUST be identified as <code>CommonAggregateComponents</code> in the document name within the schema header.
[SSM11]	A schema module defining all UBLCommon Basic Components MUST be created.
[SSM12]	The UBL Common Basic Components schema module MUST be identified as <code>CommonBasicComponents</code> in the document name within the schema header.
[SSM18]	A schema module defining all UBL Qualified Datatypes MUST be created.
[SSM19]	The UBL Qualified Datatypes schema module MUST be identified as <code>QualifiedDatatypes</code> in the document name in the schema header.
[SSM20]	The UBL Qualified Datatypes schema module MUST import the <code>ccts:UnQualifiedDatatypes</code> schema module.
SSM21	The UBL extensions schema module MUST be identified as <code>CommonExtensionComponents</code> in the document name within the schema header.
<b>A.16 Standards Adherence rules</b>	

1624

<b>A.17 Versioning rules</b>	
[VER1]	Every UBL Schema and schema module major version committee draft MUST have an RFC 3121 document-id of the form  <name>-<major>[.<revision>]
[VER2]	Every UBL Schema and schema module major version OASIS Standard MUST have an RFC 3121 document-id of the form  <name>-<major>
[VER3]	Every minor version release of a UBL schema or schema module committee draft

	<p>MUST have an RFC 3121 document-id of the form</p> <pre>&lt;name&gt;-&lt;major&gt;[.&lt;revision&gt;]</pre>
[VER4]	<p>Every minor version release of a UBL schema or schema module OASIS Standard MUST have an RFC 3121 document-id of the form</p> <pre>&lt;name&gt;-&lt;major &gt;</pre>
[VER5]	<p>For UBL Minor version changes the namespace name MUST not change</p>
[VER6]	<p>Every UBL Schema and schema module major version number MUST be a sequentially assigned, incremental number greater than zero.</p>
[VER7]	<p>Every UBL Schema and schema module minor version number MUST be a sequentially assigned, incremental non-negative integer.</p>
f[VER10]	<p>UBL Schema and schema module minor version changes MUST not break semantic compatibility with prior versions.</p>
[VER11]	<p>Every UBL Schema and schema module major version committee draft MUST capture its version number in the <code>xsd:version</code> attribute of the <code>xsd:schema</code> element in the form</p> <pre>&lt;major&gt;.0[.&lt;revision&gt;]</pre>
[VER12]	<p>Every UBL Schema and schema module major version OASIS Standard MUST capture its version number in the <code>xsd:version</code> attribute of the <code>xsd:schema</code> element in the form</p> <pre>&lt;major&gt;.0</pre>
[VER13]	<p>Every minor version release of a UBL schema or schema module committee draft MUST capture its version information in the <code>xsd:version</code> attribute in the form</p> <pre>&lt;major&gt;.&lt;non-zero&gt;[.&lt;revision&gt;]</pre>
[VER14]	<p>Every minor version release of a UBL schema or schema module OASIS Standard MUST capture its version information in the <code>xsd:version</code> attribute in the form</p> <pre>&lt;major&gt;.&lt;non-zero&gt;</pre>
[VER15]	<p>Every UBL document schema MUST declare an optional element named "UBLVersionID" immediately following the optional 'UBL Extensions' element.</p>

1625

## Appendix B. Technical Terminology

1626

Ad hoc schema processing	Doing partial schema processing, but not with official schema validator software; e.g., reading through schema to get the default values out of it.
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.
Application-level validation	Adherence to business requirements, such as valid account numbers.
Assembly	Using parts of the library of reusable UBL components to create a new kind of business document type.
Business Context	<p>Defines a context in which a business has chosen to employ an information entity.</p> <p>The formal description of a specific business circumstance as identified by the values of a set of <i>Context Categories</i>, allowing different business circumstances to be uniquely distinguished.</p>
Business Object	<p>An unambiguously identified, specified, referenceable, registerable and re-useable scenario or scenario component of a business transaction.</p> <p>The term business object is used in two distinct but related ways, with slightly different meanings for each usage:</p> <p>In a business model, business objects describe a business itself, and its business context. The business objects capture business concepts and express an abstract view of the business's "real world". The term "modeling business object" is used to designate this usage.</p> <p>In a design for a software system or in program code, business objects reflects how business concepts are represented in software. The abstraction here reflects the transformation of business ideas into a software realization. The term "systems business objects" is used to designate this usage.</p>

Business semantic(s)	A precise meaning of words from a business perspective.
Business Term	This is a synonym under which the Core Component or Business Information Entity is commonly known and used in the business. A Core Component or Business Information Entity may have several business terms or synonyms.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. See interface.
Class diagram	Shows static structure of concepts, types, and classes. Concepts show how users think about the world; types show interfaces of software components; classes show implementation of software components. (OMG Distilled)  A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships. (Rational Unified Process)
Classification scheme	This is an officially supported scheme to describe a given <i>Context Category</i>
Common attribute	An attribute that has identical meaning on the multiple elements on which it appears. A common attribute might or might not correspond to an XSD global attribute.
Component	One of the individual entities contributing to a whole.
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. (See Business Context.)
Context category	A group of one or more related values used to express a characteristic of a business circumstance.
Document schema	A schema document corresponding to a single namespace, which is likely to pull in (by including or importing) schema modules.
Core Component	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 Type	A Core Component which consists of one and only one Content Component that carries the actual content plus one or more Supplementary Components giving an essential extra definition to the Content Component. Core Component Types do not have business semantics.
Datatype	<p>A descriptor of a set of values that lack identity and whose operations do not have side effects. Datatypes include primitive pre-defined types and user-definable types. Pre-defined types include numbers, string and time. User-definable types include enumerations. (XSD)</p> <p>Defines the set of valid values that can be used for a particular <i>Basic Core Component Property</i> or <i>Basic Business Information Entity Property</i>. It is defined by specifying restrictions on the <i>Core Component Type</i> that forms the basis of the <i>Datatype</i>. (CCTS)</p>
Generic BIE	A semantic model that has a “zeroed” context. We are assuming that it covers the requirements of 80% of business uses, and therefore is useful in that state.
Instance	An individual entity satisfying the description of a class or type.
Instance constraint checking	Additional validation checking of an instance, beyond what XSD makes available, that relies only on constraints describable in terms of the instance and not additional business knowledge; e.g., checking co-occurrence constraints across elements and attributes. Such constraints might be able to be described in terms of Schematron.
Instance root/doctype	This is still mushy. The transitive closure of all the declarations imported from whatever namespaces are necessary. A doctype may have several namespaces used within it.
Intermediate element	An element not at the top level that is of a complex type, only containing other elements and attributes.
Internal schema module:	A schema module that does not declare a target namespace.

Leaf element	An element containing only character data (though it may also have attributes). Note that, because of the XSD mechanisms involved, a leaf element that has attributes must be declared as having a complex type, but a leaf element with no attributes may be declared with either a simple type or a complex type.
Lower-level element	An element that appears inside a business message. Lower-level elements consist of intermediate and leaf level.
Object Class	The logical data grouping (in a logical data model) to which a data element belongs (ISO11179). The <i>Object Class</i> is the part of a <i>Core Component's Dictionary Entry Name</i> that represents an activity or object in a specific <i>Context</i> .
Namespace schema module:	A schema module that declares a target namespace and is likely to pull in (by including or importing) schema modules.
Naming Convention	The set of rules that together comprise how the dictionary entry name for <i>Core Components</i> and <i>Business Information Entities</i> are constructed.
(XML) Schema	An XML Schema consists of components such as type definitions and element declarations. These can be used to assess the validity of well-formed element and attribute information items (as defined in [XML-Infoset]), and furthermore may specify augmentations to those items and their descendants.
Schema module	A collection of XML constructs that together constitute an XSD conformant schema. Schema modules are intended to be used in combination with other XSD conformant schema.
Schema Processing	Schema validation checking plus provision of default values and provision of new infoset properties.
Schema Validation	Adherence to an XSD schema.
Semantic	Relating to meaning in language; relating to the connotations of words.
Top-level element	An element that encloses a whole UBL business message. Note that UBL business messages might be carried by messaging transport protocols that themselves have higher-level XML structure. Thus, a UBL top-level element is not necessarily the root element of the XML document that carries it.

Type	<p>Description of a set of entities that share common characteristics, relations, attributes, and semantics.</p> <p>A stereotype of class that is used to specify an area of instances (objects) together with the operations applicable to the objects. A type may not contain any methods. See class, instance. Contrast interface.</p>
------	---



---

1627 **Appendix C. References**

- 1628 **[CCTS]** *ISO 15000-5 ebXML Core Components Technical Specification*
- 1629 **[ISONaming]** *ISO/IEC 11179, Final committee draft, Parts 1-6.*
- 1630 **[RFC 2119]** *S. Bradner, Key words for use in RFCs to Indicate Requirement Levels,*  
1631 *<http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.*
- 1632 **[UBLChart]** *UBL TC Charter, <http://oasis-open.org/committees/ubl/charter/ubl>.*
- 1633 **[XML]** *Extensible Markup Language (XML) 1.0 (Second Edition), W3C*  
1634 *Recommendation, October 6, 2000*
- 1635 **[XSD]** *XML Schema, W3C Recommendations Parts 0, 1, and 2. 2 May 2001.*
- 1636 **[XHTML]** *XHTML™ Basic, W3C Recommendation 19 December 2000:*  
1637 *<http://www.w3.org/TR/2000/REC-xhtml-basic-20001219>*

---

1638

## Appendix D. Notices

1639 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
1640 that might be claimed to pertain to the implementation or use of the technology described in this  
1641 document or the extent to which any license under such rights might or might not be available;  
1642 neither does it represent that it has made any effort to identify any such rights. Information on  
1643 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
1644 website. Copies of claims of rights made available for publication and any assurances of licenses  
1645 to be made available, or the result of an attempt made to obtain a general license or permission  
1646 for the use of such proprietary rights by implementors or users of this specification, can be  
1647 obtained from the OASIS Executive Director.

1648 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
1649 applications, or other proprietary rights which may cover technology that may be required to  
1650 implement this specification. Please address the information to the OASIS Executive Director.

1651 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]  
1652 2006. All Rights Reserved.

1653 This document and translations of it may be copied and furnished to others, and derivative works  
1654 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
1655 published and distributed, in whole or in part, without restriction of any kind, provided that the  
1656 above copyright notice and this paragraph are included on all such copies and derivative works.  
1657 However, this document itself does not be modified in any way, such as by removing the  
1658 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
1659 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
1660 Property Rights document must be followed, or as required to translate it into languages other  
1661 than English.

1662 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
1663 successors or assigns.

1664 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1665 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
1666 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE  
1667 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
1668 PARTICULAR PURPOSE.