

1

2

# **XML Naming and Design Rules For CCTS 2.01**

3

4

**Version 2.1.1**

5

**31 August 2021**

6

7 **1 Status of this Document**

8 This Technical Specification is being developed in accordance with the  
9 ECE/TRADE/C/CEFACT/2010/24/Rev.2.3 Open Development Process.

10 This document is for publication.

11

## 12 **2 UN/CEFACT XML Naming and Design Rules Project**

### 13 **Team Participants**

14 We would like to recognise the following for their significant participation to the development of this  
15 Technical Specification.

16 Project Team Leader:

17 Chris Hassler

18 Lead Editor:

19 Mary Kay Blantz

20 Contributors:

21 Sylvia Webb

22 Karina Duvinger

23 Kevin Smith

24 Ewa Iwicka

25 Sue Probert

26 Michael Dill

27 Gerhard Heemskerk

28 Hidekazu Enjo

29 Hisanao Sugamata

30 Contributors to Previous Versions

31 Mark Crawford

32 Paula Heilig

33 Gunter Stuhec

34 Garret Minikawa

35 Hisanao Sugamata

36 Frank Lin

37 K.K. Suen

38 Luc Mouchot

39 Thomas Bikeev

40 Jostein Frømyr

41 Sue Probert

42 Alain Dechamps

43 Michael Dill

44

45	<b>3 Table of Contents</b>	
46	1 Status of this Document .....	2
47	2 UN/CEFACT XML Naming and Design Rules Project Team Participants.....	3
48	4 Introduction.....	9
49	4.1 Scope and Focus .....	9
50	4.2 Audience.....	9
51	4.3 Structure of this Specification .....	9
52	4.4 Terminology and Notation .....	10
53	4.5 Related Documents .....	10
54	4.6 Conformance .....	10
55	4.7 Guiding Principles.....	10
56	5 General XML Construct.....	12
57	5.1 Overall Schema Structure.....	12
58	5.2 Relationship to CCTS.....	12
59	5.2.1 CCTS.....	12
60	5.2.2 Business Information Entities.....	13
61	5.2.3 The XML Constructs.....	14
62	5.3 Naming and Modelling Constraints .....	16
63	5.3.1 Element Naming Conventions .....	18
64	5.4 Reusability Scheme (Informative) .....	18
65	5.5 Modularity Model.....	19
66	5.5.1 Root Schema.....	21
67	5.5.2 Internal Schema.....	22
68	5.5.3 External Schema .....	22
69	5.6 Namespace Scheme .....	25
70	5.6.1 Namespace Scheme.....	26
71	5.6.2 Declaring Namespace .....	26
72	5.6.3 Namespace Persistence.....	27
73	5.6.4 Namespace Uniform Resource Identifiers.....	27
74	5.6.5 Namespace Constraint .....	28
75	5.6.6 UN/CEFACT XSD Namespace Schema Tokens.....	28
76	5.7 Schema Location.....	28
77	5.8 Versioning.....	29
78	5.8.1 Major Versions.....	29

79	5.8.2	Minor Versions .....	29
80	6	General XML Schema Language Conventions .....	31
81	6.1	Schema Construct.....	31
82	6.1.1	Constraints on Schema Construction .....	31
83	6.2	Attribute and Element Declarations.....	31
84	6.2.1	Attributes.....	31
85	6.2.2	Elements .....	32
86	6.3	Type Declarations .....	32
87	6.3.1	Usage of Types.....	32
88	6.3.2	Simple Type Definitions.....	33
89	6.3.3	Complex Type Definitions.....	33
90	6.4	User of XSD Extension and Restriction .....	33
91	6.4.1	Extension .....	34
92	6.4.2	Restriction.....	34
93	6.5	Annotation.....	34
94	6.5.1	Documentation.....	34
95	7	XML Schema Modules .....	38
96	7.1	Root Schema.....	38
97	7.1.1	Schema Construct.....	38
98	7.1.2	Namespace Scheme.....	39
99	7.1.3	Imports and Includes.....	39
100	7.1.4	Root Element Declaration .....	40
101	7.1.5	Type Definitions.....	40
102	7.1.6	Annotations .....	41
103	7.2	Internal Schema.....	41
104	7.2.1	Schema Construct.....	41
105	7.2.2	Namespace Scheme.....	41
106	7.2.3	Imports and Includes .....	41
107	7.3	Reusable Aggregate Business Information Entity Schema .....	42
108	7.3.1	Schema Construct.....	42
109	7.3.2	Namespace Scheme.....	42
110	7.3.3	Imports and Includes.....	42
111	7.3.4	Type Declarations .....	43
112	7.3.5	Element Declarations and References.....	45

113	7.3.6	Annotation.....	46
114	7.4	Core Component Type.....	49
115	7.4.1	Use of Core Component Type Module.....	49
116	7.4.2	Schema Construct.....	49
117	7.4.3	Namespace Scheme.....	50
118	7.4.4	Imports and Includes.....	50
119	7.4.5	Type Definitions.....	50
120	7.4.6	Attribute Declarations.....	51
121	7.4.7	Extension and Restriction.....	51
122	7.4.8	Annotation.....	51
123	7.5	Unqualified Data Type.....	52
124	7.5.1	Use of Unqualified Data Type Module.....	52
125	7.5.2	Schema Construct.....	53
126	7.5.3	Namespace Scheme.....	53
127	7.5.4	Imports and Includes.....	53
128	7.5.5	Type Definitions.....	54
129	7.5.6	Attribute Declarations.....	54
130	7.5.7	Extension and Restriction.....	57
131	7.5.8	Annotation.....	57
132	7.6	Qualified Data Type.....	58
133	7.6.1	Use of Qualified Data Type Schema Module.....	58
134	7.6.2	Schema Construct.....	58
135	7.6.3	Namespace Scheme.....	59
136	7.6.4	Imports and Includes.....	59
137	7.6.5	Type Definitions.....	59
138	7.6.6	Attribute and Element Declarations.....	61
139	7.6.7	Annotation.....	62
140	7.7	Code Lists.....	63
141	7.7.1	Schema Construct.....	64
142	7.7.2	Namespace Name for Code Lists.....	65
143	7.7.3	UN/CEFACT XSD Schema Namespace Token for Code Lists.....	66
144	7.7.4	Schema Location.....	67
145	7.7.5	Imports and Includes.....	68
146	7.7.6	Type Definitions.....	68

147	7.7.7	Element and Attribute Declarations.....	69
148	7.7.8	Extension and Restriction.....	69
149	7.7.9	Annotation.....	69
150	7.8	Identifier List Schema.....	70
151	7.8.1	Schema Construct.....	70
152	7.8.2	Namespace Name For Identifier List Schema.....	71
153	7.8.3	UN/CEFACT XSD Namespace Token for Identifier List Schema.....	72
154	7.8.4	Schema Location.....	72
155	7.8.5	Imports and Includes.....	73
156	7.8.6	Type Definitions.....	73
157	7.8.7	Attribute and Element Declarations.....	74
158	7.8.8	Extension and Restriction.....	74
159	7.8.9	Annotation.....	75
160	8	XML Instance Documents.....	76
161	8.1	Character Encoding.....	76
162	8.2	xsi:schemaLocation.....	76
163	8.3	Empty Content.....	76
164	8.4	xsi:type.....	76
165	Appendix A	Related Documents.....	77
166	Appendix B	Overall Structure.....	78
167	B.1	XML Declaration.....	78
168	B.2	Schema Module Identification and Intellectual Property Disclaimer.....	78
169	B.3	Schema Start Tag.....	78
170	B.4	Includes.....	79
171	B.5	Imports.....	79
172	B.6	Root Element.....	80
173	B.7	Type Definitions.....	81
174	Appendix C	BPS Approved Acronyms and Abbreviations.....	84
175	Appendix D	Common Use Cases for Code Lists and Identifier Lists.....	85
176	D.1	The Use of Code Lists within XML Schemas.....	85
177	D.1.1	Referencing a Predefined Standard Code List in and Unqualified Data Type.....	86
178	D.1.2	Referencing Any Code List Using the Unqualified Data Type udt:CodeType.....	87
179	D.1.3	Referencing a Predefined Code List by Declaring a Specific Qualified Data Type.....	87
180	D.1.4	Choosing or Combining Values from Different Code Lists.....	88

181	D.1.5	Restricting Allowed Code Values .....	89
182	D.2	The Use of Identifier Schemes within XML Schemas .....	90
183	Appendix E	Annotation Templates .....	91
184	Appendix F	Naming and Design Rules Checklist .....	95
185	Appendix G:	Glossary .....	109
186	Intellectual Property Disclaimer .....		113
187			



## 188 4 Introduction

189 This UN/CEFACT – *XML Naming and Design Rules* Technical Specification describes and  
190 specifies the rules and guidelines that will be applied by UN/CEFACT when developing XML  
191 schema.

192 This technical specification provides a way to identify, capture and maximize the reuse of business  
193 information expressed as XML schema components to support and enhance information  
194 interoperability across multiple business situations.

### 195 4.1 Scope and Focus

196 This UN/CEFACT – *XML Naming and Design Rules* Technical Specification can be employed  
197 wherever business information is being shared or exchanged amongst and between enterprises,  
198 governmental agencies, and/or other organizations in an open and worldwide environment using  
199 XML schema for defining the content of the business information payload.

200 This technical specification will form the basis for standards development work of technical experts  
201 developing XML schema based on information models developed in accordance with the  
202 UN/CEFACT *Core Components Technical Specification – Part 8 of the ebXML Framework (CCTS)*,  
203 version 2.01 plus corrigenda.

204 This version was amended from the original to correct certain errors in the original text, as well as  
205 to fulfill the following goals: (1) decoupling the unqualified data type schema from this  
206 specification, to allow for easier maintenance of the data type catalogue, (2) to enable relative  
207 path names, making implementation of the schemas easier, (3) incorporating the Core Component  
208 Business Document Assembly specification, which did not exist at the time the first version was  
209 created, and (4) selectively decoupling any or all code list and identifier value enumerations from  
210 the qualified data type, to allow for two-phase code validation processing and flexibility.

211 [Note] - The term “decoupling” refers to decoupling of unqualified data type schema from the data type  
212 catalogue (CCT) and/or decoupling a particular qualified data type from a set of value enumerations.

### 213 4.2 Audience

214 The primary audience for this UN/CEFACT – *XML Naming and Design Rules* Technical  
215 Specification are members of the UN/CEFACT Bureau Programme Support who are responsible for  
216 development and maintenance of UN/CEFACT XML schema. The intended audience also  
217 includes the wider membership of the other UN/CEFACT groups who will participate in the process  
218 of creating and maintaining UN/CEFACT XML schema.

219 Additional audiences are designers of tools who need to specify the conversion of user input into  
220 XML schema representation adhering to the rules defined in this document. Additionally, designers  
221 of XML schema outside of the UN/CEFACT Forum community may find the rules contained herein  
222 suitable as design rules for their own organization. Since the constructs defined in CCTS are  
223 consistent with UML classes, attributes, and associations, these design rules can easily be applied  
224 to non CCTS constructs as well.

### 225 4.3 Structure of this Specification

226 The UN/CEFACT *XML Naming and Design Rules* Technical Specification has been divided into 5  
227 main sections:

- 228 • Section 4 provides general information about the document itself as well as normative  
229 statements in respect to conformance, and guiding principles applied in developing this  
230 specification.
- 231 • Section 5 provides information on this specification's dependency and relationship to  
232 CCTS. Furthermore, this section describes the approach taken to modularity in order to  
233 maximize the reuse of business information expressed as XML schema components and  
234 the general naming conventions applied. (Normative, except for Section 5.4, which is  
235 Informative)

- 236 • Section 6 provides the general conventions applied with respect to the use of the XML  
237 schema language. (Normative)
- 238 • Section 7 provides detailed rules applicable to each of the schema modules defined  
239 by the modularity approach. (Normative)
- 240 • Section 8 provides guidelines and rules related to XML instance documents. (Normative)

241 The document also contains the following Appendices:

- 242 • Appendix A Related Documents (Informative)
- 243 • Appendix B Overall Structure (Normative)
- 244 • Appendix C BPS Approved Acronyms and Abbreviations (Normative)
- 245 • Appendix D Use cases for code lists and identifier lists. (Informative)
- 246 • Appendix E Annotation Templates (Informative)
- 247 • Appendix F Naming and Design Rules List (Normative)
- 248 • Appendix G Glossary (Informative)

## 249 4.4 Terminology and Notation

250 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,  
251 RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted  
252 as described in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2119.<sup>1</sup>  
253 Wherever xsd: appears this refers to a construct taken from the W3C XML schema specification.  
254 Wherever ccts: appears this refers to a construct taken from the CCTS.

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

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

257 [Rn] – Identification of a rule that requires conformance. Rules are normative. In order to ensure  
258 continuity across versions of the specification, rule numbers that are deleted will not be re-issued,  
259 and any new rules will be assigned the next higher number - regardless of location in the text.

260 *Courier* – All words appearing in **bolded courier font** are values, objects or

261 keywords. When defining rules the following annotations are used:

- 262 • [ ] = optional
- 263 • < > = Variable
- 264 • | = choice

## 265 4.5 Related Documents

266 Related documents referenced in this specification are listed in Appendix A.

## 267 4.6 Conformance

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

---

270 [R1] Conformance shall be determined through adherence to the content of normative  
271 sections, rules and definitions.

---

## 272 4.7 Guiding Principles

273 The following guiding principles were used as the basis for all design rules contained in this  
274 document:

---

<sup>1</sup> Key words for use in RFCs to Indicate Requirement Levels - Internet Engineering Task Force, Request For Comments 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>

- 275 ○ Relationship to UMM – UN/CEFACT XML Schema Definition Language (XSD) Schema will
- 276 be based on UMM metamodel adherent Business Process Models.
- 277 ○ Relationship to Information Models – UN/CEFACT XSD Schema will be based on
- 278 information models developed in accordance with the UN/CEFACT – *Core Components*
- 279 *Technical Specification*.
- 280 ○ Schema Creation– UN/CEFACT XML design rules will support schema creation
- 281 through handcrafting as well as automatic generation.
- 282 ○ Interchange and Application Use – UN/CEFACT XSD Schema and instance documents
- 283 are intended for business-to-business and application-to-application use.
- 284 ○ Tool Use and Support - The design of UN/CEFACT XSD Schema will not make any
- 285 assumptions about sophisticated tools for creation, management, storage, or presentation
- 286 being available.
- 287 ○ Legibility - UN/CEFACT XML instance documents should be intuitive and reasonably clear in
- 288 the context for which they are designed.
- 289 ○ Schema Features - The design of UN/CEFACT XSD Schema should use the most
- 290 commonly supported features of W3C XSD Schema.
- 291 ○ Technical Specifications – UN/CEFACT XML Naming and Design Rules will be based on
- 292 Technical Specifications holding the equivalent of W3C recommended status.
- 293 ○ Schema Specification – UN/CEFACT XML Naming and Design rules will be fully conformant with
- 294 W3C XML Schema Definition Language.
- 295 ○ Interoperability - The number of ways to express the same information in a UN/CEFACT
- 296 XSD Schema and UN/CEFACT XML instance document is to be kept as close to one as
- 297 possible.
- 298 ○ Maintenance – The design of UN/CEFACT XSD Schema must facilitate maintenance.
- 299 ○ Context Sensitivity - The design of UN/CEFACT XSD Schema must ensure that context-
- 300 sensitive document types are not precluded.
- 301 ○ Relationship to Other Namespaces - UN/CEFACT XML design rules will be cautious about
- 302 making dependencies on other namespaces.
- 303 ○ Legacy formats - UN/CEFACT XML Naming and Design Rules are not responsible for
- 304 sustaining legacy formats.
- 305

## 306 5 General XML Construct

307 This section defines rules related to general XML constructs to include:

- 308
- 309 ○ Overall Schema Structure
- 310 ○ Relationship to CCTS
- 311 ○ Naming and Modelling Constraints
- 312 ○ Reusability Scheme
- 313 ○ Modularity Model
- 314 ○ Namespace Scheme
- 315 ○ Schema Location
- 316 ○ Versioning Scheme

### 317 5.1 Overall Schema Structure

318 UN/CEFACT has determined that the World Wide Web Consortium (W3C) XML schema definition  
319 (XSD) language is the generally accepted schema language experiencing the broadest adoption.  
320 Accordingly, all UN/CEFACT normative schema will be expressed in XSD. All references to XML  
321 schema will be as XSD schema or UN/CEFACT XSD Schema.

---

322 [R2] All UN/CEFACT XSD Schema design rules MUST be based on the *W3C XML Schema*  
323 *Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Data*  
324 *Types*

---

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

---

328 [R3] All UN/CEFACT XSD Schema and UN/CEFACT conformant XML instance documents  
329 MUST be based on the W3C suite of technical specifications holding recommendation  
330 status.

---

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

---

333 [R4] UN/CEFACT XSD Schema MUST follow the standard structure defined in Appendix B.

---

### 334 5.2 Relationship to CCTS

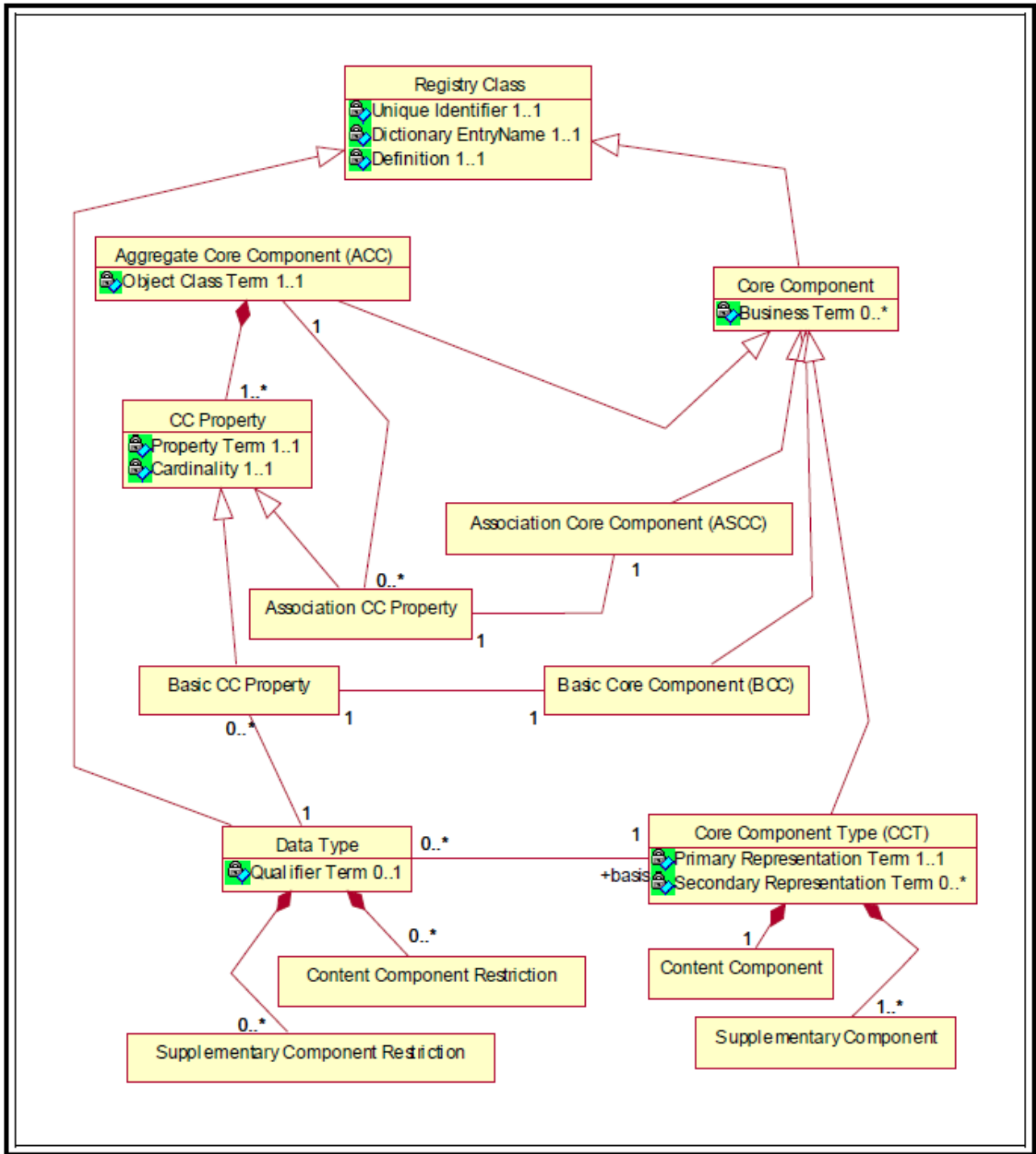
335 All UN/CEFACT business information modelling and business process modelling  
336 employ the methodology and model described in CCTS.

#### 337 5.2.1 CCTS

338 CCTS defines context neutral and context specific information building blocks. Context neutral  
339 information components are defined as Core Components (**ccts:CoreComponents**). Context  
340 neutral **ccts:CoreComponents** are defined in CCTS as “A building block for the creation of a  
341 semantically correct and meaningful information exchange package. It contains only the  
342 information pieces necessary to describe a specific concept.”<sup>2</sup> Figure 5-1 illustrates the various  
343 pieces of the overall **ccts:CoreComponents** metamodel.

---

<sup>2</sup> *Core Components Technical Specification, Part 8 of the ebXML Technical Framework Version 2.01 (Second Edition), UN/CEFACT, 15 November 2003, plus corrigenda*



345 **Figure 5-1 Core Component Metamodel**

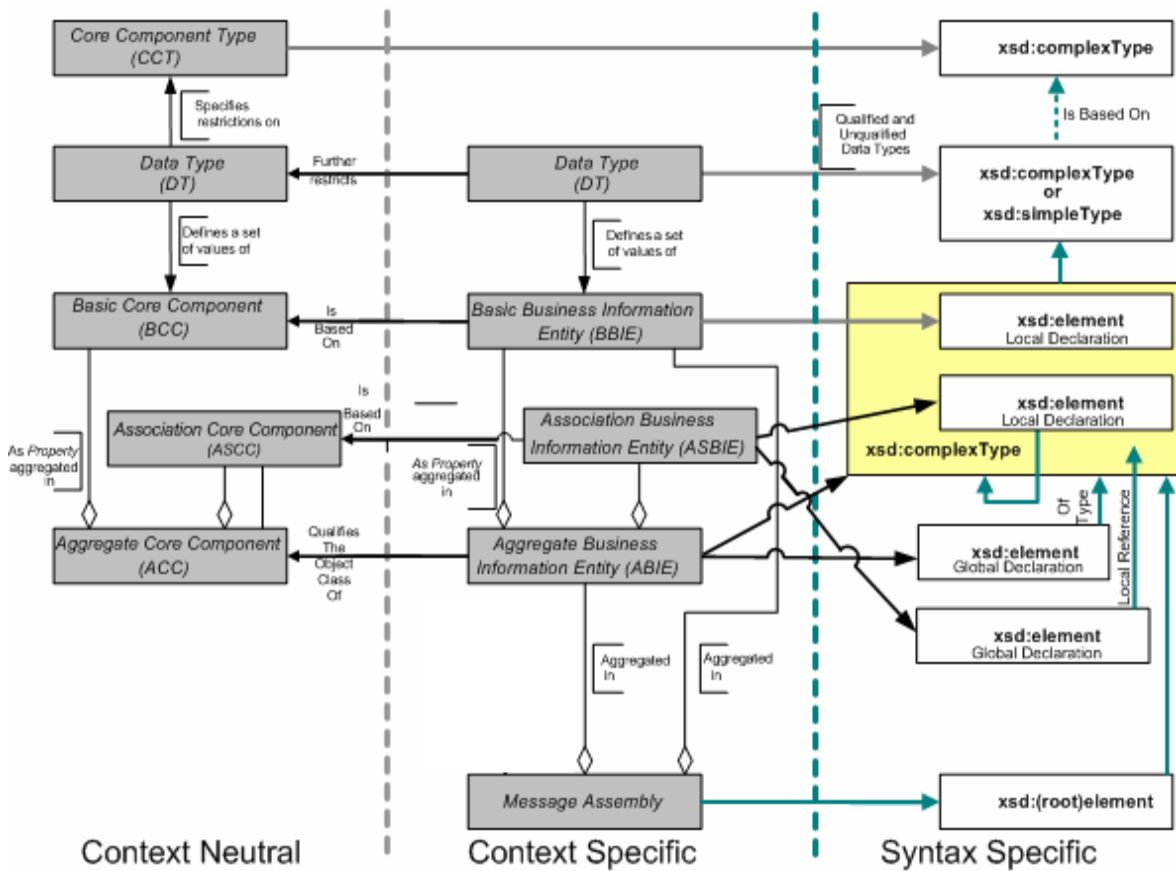
346 **5.2.2 Business Information Entities**

347 In the CCTS model, context neutral core components are instantiated as context specific  
 348 components for business information payload and model harmonization. The context specific  
 349 components are defined as Business Information Entities. (See CCTS Section 6.2 for a detailed  
 350 discussion of the UN/CEFACT context mechanism.)<sup>3</sup> Context specific CCTS Business

<sup>3</sup> *Core Components Technical Specification, Part 8 of the ebXML Technical Framework Version 2.0 (Second Edition)*, UN/CEFACT, 15 November 2003



363 Business Information Entities), and the other boxes reflect XSD constructs (`xsd:type`,  
 364 `xsd:element`, `xsd:attribute`). The relationships follow the following basic principles:



365

366 **Figure 5-3 Relationship between CCTS and XSD Artefacts in UN/CEFACT XSD**  
 367 **Schema**

- 368 ○ The business information payload (Message Assembly) is represented as a  
 369 `xsd:complexType` definition and global element declaration in an UN/CEFACT XSD  
 370 Schema. The global element declaration is based on (is of type) `xsd:complexType` that  
 371 represents the document level ABIE. The global element appears in, and is designated as the  
 372 root element of, UN/CEFACT conformant XML instances.
- 373 ○ An ABIE is defined as a `xsd:complexType` and a corresponding global  
 374 `xsd:element` is declared.
- 375 ○ Depending on the type of association, an ASBIE will be declared as either a local element or  
 376 as a global element. If the ASBIE is a composition it will be declared as a local element within  
 377 the `xsd:complexType` representing the associating ABIE. If it is not a composition (i.e.,  
 378 aggregation) the ASBIE is included in the content model by referencing the global element that  
 379 was declared for the associated ABIE. The ASBIE element is in itself based on (is of type)  
 380 `xsd:complexType` of the associated ABIE. In this way the content model of the associated  
 381 ABIE is included in the content model of the associating ABIE.

382

[Note]

383 Per CCTS, an ABIE can contain other ABIEs in ever higher levels of  
 384 aggregation. When an ABIE contains another ABIE, this is accomplished  
 385 through the use of ASBIEs. The ASBIE is the linking mechanism that shows the  
 386 hierarchical relationship between ABIE constructs. When an ASBIE is used, we  
 387 refer to the ABIE that contains it as the associating ABIE, and the ABIE that it  
 388 represents as the associated ABIE.

- 389 ○ A BBIE is declared as a local element within the `xsd:complexType` representing the parent  
 390 ABIE. The BBIE is based on a (is of type) qualified or unqualified data type (DT).



- 391 ○ A DT is defined as either a `xsd:complexType` or `xsd:simpleType`. DT's are based on  
392 Core Component Type `xsd:complexType` from the Core Component Type (CCT) schema  
393 module. These data types can be unqualified (no additional restrictions above those  
394 imposed by the CCT type) or qualified (additional restrictions above those imposed by the  
395 CCT type). XSD built-in data types will be used whenever the facets of the built-in data type  
396 are equivalent to the CCT supplementary components for that data type.

397 [Note]

398 Data Types are not derived from the CCT complex types using `xsd:restriction`  
399 because whereas all CCTs are defined as complex types with attributes representing their  
400 supplementary components, in some cases built-in XSD data types whose facets  
401 correspond to the supplementary components are leveraged . See Section 7.5 for more  
402 information.

- 403 ○ A CCT is defined as a `xsd:complexType`. Supplementary components are declared as  
404 attributes for the CCT `xsd:complexType`. CCTs are contained in the Core Component  
405 Type Schema Module which is considered the normative XSD expression of CCTS Core  
406 Component Type.

### 407 5.3 Naming and Modelling Constraints

408 UN/CEFACT XSD Schema are derived from components created through the application of CCTS,  
409 UN/CEFACT Modelling Methodology (UMM) process modelling and data analysis, and Core  
410 Component Business Document Assembly (CCBDA). UN/CEFACT XSD Schema contain XML  
411 syntax specific constructs that follow the naming and design rules in this specification. Those  
412 naming and design rules have taken advantage of the features of XSD to incorporate naming  
413 constraint rules that in many cases result in truncation of the CCTS dictionary entry names.  
414 However, the fully conformant CCTS dictionary entry names of the underlying CCTS registry artefact  
415 are preserved as part of the `xsd:<annotation>` element accompanying each element declaration  
416 in UN/CEFACT schema, and can be reconstructed through use of XPath expressions. The XML  
417 fully qualified XPath ties the information to its standardized semantics as described in the underlying  
418 CCTS construct and CCTS dictionary entry name, while the XML element or attribute name is a  
419 truncation that reflects the hierarchy inherent in the XML construct. There are differences in the rules  
420 for naming of elements, attributes, and types.

---

421 [R5] Each element or attribute XML name MUST have one and only one fully qualified  
422 XPath(FQXP)

---

423 This rule and the other rules on element naming imply that a part of the fully qualified XPath will  
424 always represent the CCTS dictionary entry name of the corresponding ABIE, BBIE, ASBIE or DT.

#### 425 Example 5-1: Fully Qualified XPath

```
426 /CrossIndustryInvoice/CIExchangedDocumentContext/SpecifiedTransaction/Identifier  
427 /Tender/ProcuringOrganization/Name/Text
```

428 The official language for UN/CEFACT is English. All official XML constructs as published by  
429 UN/CEFACT will be in English. XML development work may very well occur in other languages,  
430 however official submissions for inclusion in the UN/CEFACT XML library must be in English. Other  
431 language translations of UN/CEFACT published XML components are at the discretion of users.

---

432 [R6] Element, attribute and type names MUST be composed of words in the English  
433 language, using the primary English spellings provided in the Oxford English  
434 Dictionary.

---

435 Following the *ebXML Architecture Specification* and commonly used best practice, Lower Camel  
436 Case (LCC) is used for naming attributes and Upper Camel Case (UCC) is used for naming  
437 elements and types. Lower Camel Case capitalizes the first character of each word except the first  
438 word and compounds the name. Upper Camel Case capitalizes the first character of each word and  
439 compounds the name.

---

440 [R7] Lower camel case (LCC) MUST be used for naming attributes  
441

---



442 **Example 5-2: Attribute**

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

[R8] Upper camel case (UCC) MUST be used for naming elements and types.

446 **Example 5-3: Element**

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

448 **Example 5-4: Type**

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

[R9] Element, attribute and type names MUST be in singular form unless the concept itself is plural.

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

454 **Allowed - Singular:**

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

456 **Not Allowed - Plural:**

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

[R10] Element, attribute and type names MUST be drawn from the following character set: **a-z** and **A-Z**. Any special characters such as spaces, underscores, and periods that exist in the underlying Dictionary Entry Names MUST be removed.

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

463 **Not Allowed**

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

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

[R11] This rule has been combined with [R10].

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

470 **Not Allowed**

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

[R12] XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other word truncations, except those included in the UN/CEFACT controlled vocabulary or listed in Appendix C.

[R13] The acronyms and abbreviations listed in Appendix C MUST always be used.

[R14] Acronyms and abbreviations at the beginning of an attribute declaration MUST appear in all lower case. All other acronym and abbreviation usage in an attribute declaration must appear in upper case.

[R15] Acronyms MUST appear in all upper case for all element declarations and type definitions.

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

483 **Allowed – ID is an approved abbreviation**

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

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

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

### 488 5.3.1 Element Naming Conventions

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

## 494 5.4 Reusability Scheme (Informative)

495 UN/CEFACT is committed to transitioning to an object based approach for its process models  
496 and core component implementation efforts as supported in both UMM and CCTS. UN/CEFACT  
497 deliberated adopting a type based approach (named types), a type and element based approach,  
498 or an element based approach.

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

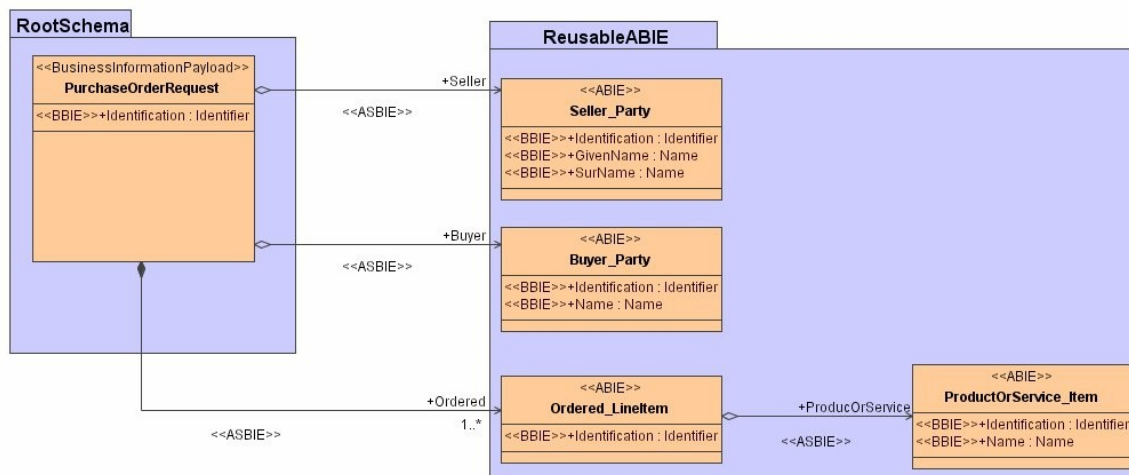
511 A purely type based approach does, however, limit the ability to reuse elements, especially in  
512 technologies such as Web Services Description Language (WSDL). UN/CEFACT has thus decided  
513 to implement what is known as a “hybrid approach” as this provides benefits over a purely type  
514 based approach. Most significantly it increases reusability of library content both at the modelling  
515 and xsd level.

516 The key principles of the “hybrid approach” are:

- 517 • All classes (PurchaseOrderRequest, Seller\_Party, Buyer\_Party, Ordered\_LineItem and  
518 ProductOrService\_Item in figure 5-4) are declared as a **xsd:complexType**.
- 519 • All attributes of a class are declared as a local **xsd:element** within a **xsd:complexType**.
- 520 • Composition associations (e.g. PurchaseOrderRequest. Ordered. Ordered\_LineItem in figure  
521 5-4) are locally declared as a **xsd:element** within a **xsd:complexType**. A composition  
522 ASBIE is defined as a specialized type of ASBIE that represents a composition relationship  
523 between the associating ABIE and the associated ABIE.
- 524 • Associations that are not defined as composites (e.g. PurchaseOrderRequest.Buyer.  
525 Buyer\_Party, PurchaseOrderRequest. Seller. SellerParty in figure 5-4) are globally  
526 declared as a **xsd:element**.

527 The rules pertaining to the ‘hybrid approach’ are contained in sections 7.3.4 and 7.3.5 for  
528 type and element declaration.

529 Figure 5-4 shows an example UML model and example 5-9 shows the resulting XSD declarations.



531 **Figure 5-4 UML model example**

532

533

**Example 5-9: xsd declarations representing Figure 5-4**

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

```

<xsd:element name="PurchaseOrderRequest" type="rsm:PurchaseOrderRequestType"/>
<xsd:element name="BuyerParty" type="ram:BuyerPartyType"/>
<xsd:element name="OrderedLineItem" type="ram:OrderedLineItemType"/>
<xsd:element name="ProductOrServiceItem" type="ram:ProductOrServiceItemType"/>
<xsd:element name="SellerParty" type="ram:SellerPartyType"/>
<xsd:complexType name="PurchaseOrderRequestType">
  <xsd:sequence
    ce>
    <xsd:element name="ID" type="udt:IDType"/>
    <xsd:element ref="ram:SellerParty"/>
    <xsd:element ref="ram:BuyerParty"/>
    <xsd:element name="OrderedLineItem"
type="ram:OrderedLineItemType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="BuyerPartyType">
  <xsd:sequence
    ce>
    <xsd:element name="ID" type="udt:IDType"/>
    <xsd:element name="Name" type="udt:NameType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="OrderedLineItemType">
  <xsd:sequence
    ce>
    <xsd:element name="ID" type="udt:IDType"/>
    <xsd:element name="ProductOrServiceItem" type="ram:ProductOrServiceItemType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ProductOrServiceItemType">
  <xsd:sequence
    ce>
    <xsd:element name="ID" type="udt:IDType"/>
    <xsd:element name="Name" type="udt:NameType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SellerPartyType">
  <xsd:sequence
    ce>
    <xsd:element name="ID" type="udt:IDType"/>
    <xsd:element name="GivenName" type="udt:NameType"/>
    <xsd:element name="Surname" type="udt:NameType"/>
  </xsd:sequence>
</xsd:complexType>
  
```

578

## 5.5 Modularity Model

579

Modularity in schema design promotes reuse and provides significant management capabilities.

580

Modules can be either unique in their functionality, or represent splitting of larger schema files for

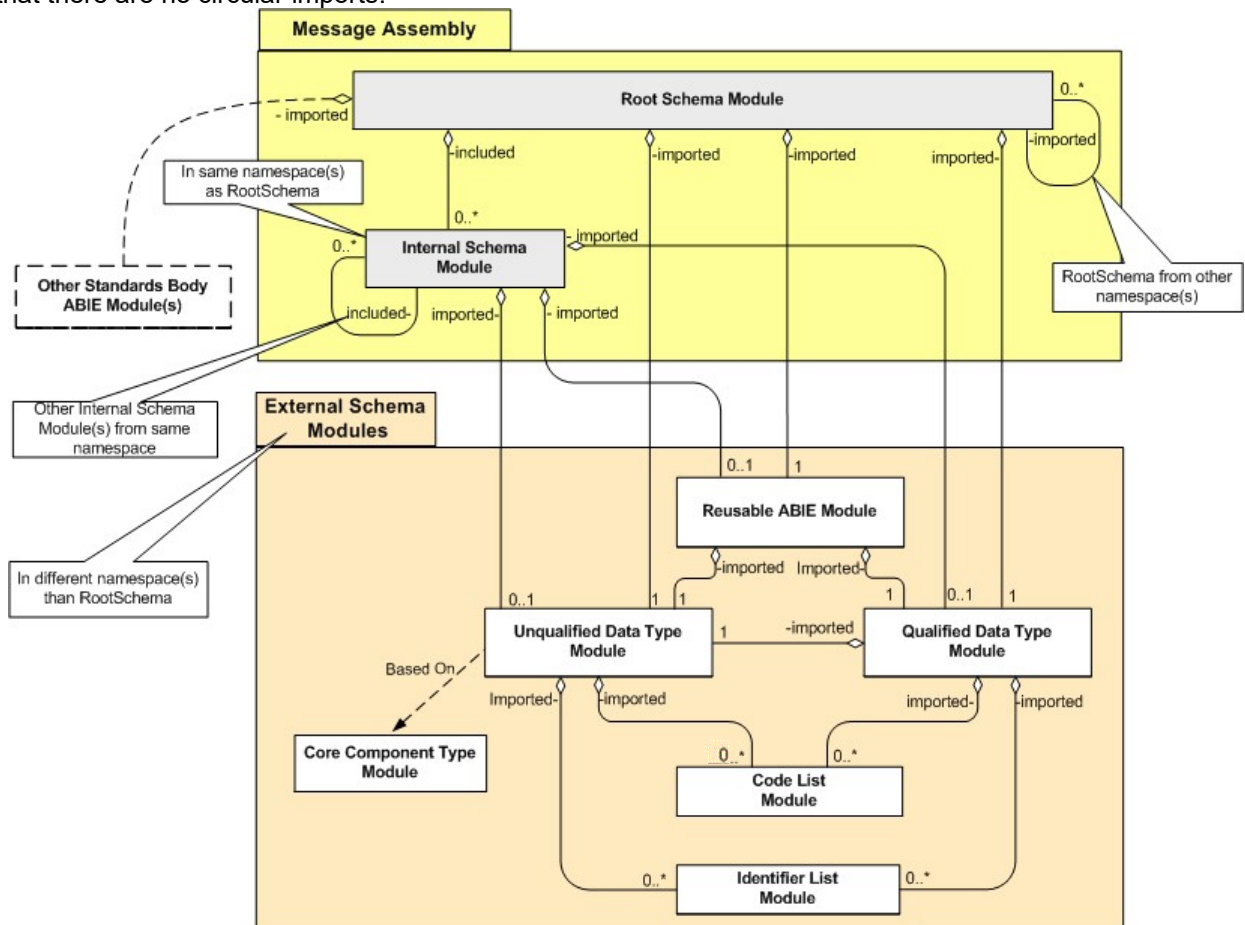
581 performance or manageability enhancement. A modularity model provides an efficient and effective  
 582 mechanism for importing and including components as needed rather than dealing with complex,  
 583 multi-focused schema.

584 Accordingly UN/CEFACT has defined a number of schema modules to support this approach.  
 585 Figure 5-5 portrays the UN/CEFACT modularity model. UN/CEFACT categorizes modules into  
 586 business information payload(s) and external schema modules. The business information payload  
 587 consists of root schema and internal schema modules that reside in the same namespace as the  
 588 root schema. The external schema modules consist of a set of reusable schema for ABIEs,  
 589 unqualified data types, qualified data types, code lists and identifier lists. Each of these schema  
 590 modules resides in its own namespace. Dependencies exist amongst the various modules as shown  
 591 in figure 5-5.

592 The root schema module always includes any internal schema residing in its namespace. It also  
 593 always imports the ABIE reusable, unqualified and qualified data type schema modules. It may  
 594 import root schemas from other namespaces as well as reusable schema from other standards  
 595 bodies. The internal schema module may include other internal schema modules from its own  
 596 namespace, and may reference – through the root schema module– other root schema modules and  
 597 their internal schema modules. It may also import the unqualified data type, qualified data type, and  
 598 reusable ABIE schema modules.

599 The reusable ABIE schema module always imports the unqualified data type and qualified data type  
 600 schema modules. The unqualified data type schema imports necessary code list schema modules  
 601 and may import identifier list schema modules. The qualified data type schema modules always  
 602 import the unqualified data type schema module as well as necessary code list and identifier list  
 603 schema modules.

604 The core component type schema module is provided as reference documentation and is used as  
 605 the basis for the unqualified data type schema module. The modularity approach has been designed  
 606 so that there are no circular imports.



607

608 **Figure 5-5 UN/CEFACT XSD Schema Modularity Scheme**

609 To ensure consistency, and for standardization of namespace tokens as addressed elsewhere in  
 610 this specification, all schema modules identified above are referred to by their formal name or  
 611 token value in the table below:

Schema Module Name	Token
Root Schema Schema Module	rsm
Core Component Type Schema Module	cct
Reusable Aggregate Business Information Entity Schema Module	ram
Unqualified Data Type Schema Module	udt
Qualified Data Type Schema Module	qdt
Code List Schema Module	clm
Identifier List Schema Module	ids

- 612
- 
- 613 [R16] The schema module file name for modules other than code lists or identifier lists **MUST**  
 614 of the form `<SchemaModuleName>_<Version>.xsd`, with periods, spaces, or other  
 615 separators and the words Schema Module removed.
- 616 [R17] The schema module file name for code lists and identifier lists, **MUST** be of the form  
 617 `<AgencyName>_<ListName>_<Version>.xsd`, with periods, spaces, or other  
 618 separators removed.
- 619 [R18] In representing versioning schemes in file names, only the major version should be  
 620 included.
- 

## 621 5.5.1 Root Schema

622 UN/CEFACT incorporates a modularity concept that leverages the benefits previously described. In  
 623 the UN/CEFACT XML repository, there are a number of UN/CEFACT root schema, each of which  
 624 expresses a separate business function.

- 
- 625 [R19] A root schema **MUST** be created for each unique business information payload.
- 

626 To ensure uniqueness, root schema modules will be given unique names that reflect the business  
 627 function being addressed by the schema. This business function is described in the UN/CEFACT  
 628 Requirements Specification Mapping (RSM) document as the target business information payload.  
 629 Accordingly, the business information payload name representing the business function will form the  
 630 basis for the root schema name.

- 
- 631 [R20] Each UN/CEFACT root schema module **MUST** be named  
 632 `<BusinessInformationPayload> Schema Module`.
- 

633 The UN/CEFACT modularity approach enables the reuse of individual root schema without having  
 634 to import the entire UN/CEFACT root schema library. Additionally, a root schema can import  
 635 individual modules without having to import all UN/CEFACT XSD schema modules. Each root  
 636 schema will define its own dependencies. A root schema should not duplicate reusable XML  
 637 constructs contained in other schema, rather it should reuse existing constructs available  
 638 elsewhere. Specifically, root schema will import or include other schema modules to maximize  
 639 reuse through `xsd:include` or `xsd:import` as appropriate.

- 
- 640 [R21] A root schema **MUST NOT** replicate reusable constructs available in schema modules  
 641 capable of being referenced through `xsd:include` or `xsd:import`.
- 

642 Schema modules used by the root schema need to be treated as either internal or external schema  
 643 modules so correct namespace decisions can be made.

- 
- 644 [R22] UN/CEFACT XSD schema modules **MUST** either be treated as external schema  
 645 modules, or as internal schema modules of the root schema.
-

## 646 5.5.2 Internal Schema

647 The Core Component Business Document Assembly (CCBDA) specification provides a mechanism  
648 for restricting ABIEs in order to assemble a single message. Messages in an XML context  
649 correspond to a root schema, and as such, the restricted ABIEs would be declared in an internal  
650 schema. These ABIEs will be defined as **xsd:complexType** in an internal schema module rather  
651 than in the reusable ABIE schema module, (See Section 5.5.3.4 below). UN/CEFACT XSD  
652 Schema may have zero or more internal schema modules.

653 Internal schema modules will reside in the same namespace as their parent root schema. Since the  
654 internal schema reside in the same namespace as the root, the root schema uses **xsd:include** to  
655 incorporate these internal modules. The UN/CEFACT XSD schema modularity approach ensures  
656 that logical associations exist between root and internal schema modules and that individual schema  
657 modules can be reused to the maximum extent possible.

---

658 [R23] All UN/CEFACT internal schema modules MUST be in the same namespace as their  
659 corresponding **rsm:RootSchema**.

---

660 UN/CEFACT internal schema modules will have a semantically meaningful name. Internal schema  
661 module names will identify the parent root schema module, the internal schema module function,  
662 and the schema module itself.

---

663 [R24] Each UN/CEFACT internal schema module MUST be named  
664 **<ParentRootSchemaModuleName><InternalSchemaModuleFunction> Schema**  
665 **Module**

---

### 666 Example 5-10: UN/CEFACT internal schema module name

```
667 TravelReservationRequestFlightInformation
668 Where:
669 TravelReservationRequest represents the parent root schema module name
670 FlightInformation represents the internal schema module function
```

## 671 5.5.3 External Schema

672 To adhere to the principles and rules contained in Section 7, schema modules will be created for  
673 reusable components. These schema modules are referred to as external schema modules because  
674 they reside in a different namespace from the root schema. Root schema may import one or more  
675 of these external schema modules. UN/CEFACT has identified the need for the following external  
676 schema modules:

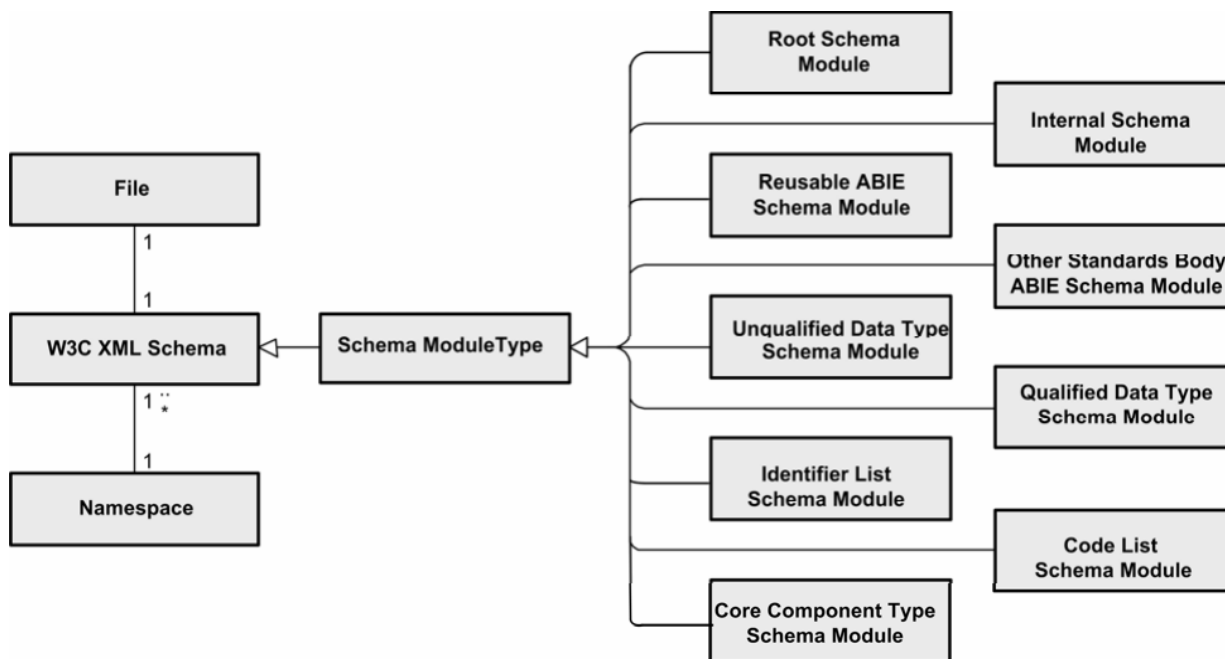
- 677 ○ Unqualified Data Type
- 678 ○ Qualified Data Type
- 679 ○ Reusable ABIE
- 680 ○ Code List
- 681 ○ Identifier List
- 682 ○ Other Standards Body ABIE module

683 [Note]

684 The terms “unqualified data type” and “qualified data type” refer to the ISO 11179  
685 concept of qualifiers for name constructs, not to the xml namespace concept of  
686 qualified and unqualified

687 These external schema modules are reflected in Figure 5-6.





689 **Figure 5-6 UN/CEFACT XSD Schema Modules**

690 **5.5.3.1 Core Component Type Schema Module**

691 A schema module is required to represent the normative form for CCTs from CCTS. This schema  
 692 module will be used as the normative reference for all CCTS based XML instantiations. This  
 693 schema will form the basis of the UDT schema module, however it will never be imported directly  
 694 into any UN/CEFACT schema module.

---

695 [R25] A Core Component Type schema module MUST be created.

---

696 The Core Component Type schema module will have a standardized name that uniquely  
 697 differentiates it from other UN/CEFACT XSD schema modules.

---

698 [R26] The `cct:CoreComponentType` schema module MUST be named 'Core Component  
 699 Type Schema Module'.

---

700 **5.5.3.2 Unqualified Data Type Schema Module**

701 A schema module is required to represent the normative form data types for each CCT as expressed  
 702 in the CCTS meta model. These data types are based on the XSD constructs from the CCT schema  
 703 module but where possible reflect the use of XSD built-in data types defined as `xsd:simpleType`  
 704 rather than their parent CCT `xsd:complexType`. As such, the unqualified data type schema  
 705 module does not import the CCT schema module.

706 An unqualified data type is defined for all approved CCTS primary and secondary representation  
 707 terms.

---

708 [R203] An Unqualified Data Type MUST NOT contain any restriction on their source CCTs other  
 709 than those defined in CCTS and agreed upon best practices.

---

710 [R27] An Unqualified Data Type schema module MUST be created

---

711 The unqualified data type schema module will have a standardized name that uniquely differentiates  
 712 it from other UN/CEFACT XSD schema modules.

---

713 [R28] The `udt:UnqualifiedDataType` schema module MUST be named 'Unqualified Data  
 714 TypeSchema Module'

---

715 **5.5.3.3 Qualified Data Type Schema Module**

716 As data types are reused for different BIEs, restrictions on the data type may be applied. These  
 717 restricted data types are referred to as qualified data types. These qualified data types will be  
 718 defined in a separate qualified data type schema module. The qualified data type schema module

719 will import the Unqualified Data Type Schema Module. In the future, this single qualified data type  
720 schema module may be segmented into additional modules if deemed necessary.

---

721 [R29] A Qualified Data Type schema module MUST be created.

---

722 The qualified data type schema module will have a standardized name that uniquely differentiates  
723 it from other UN/CEFACT XSD schema modules.

---

724 [R30] The `qdt:QualifiedDataType` schema module MUST be named 'Qualified Data Type  
725 Schema Module'.

---

#### 726 **5.5.3.4 Reusable Aggregate Business Information Entity Schema Module**

727 A single reusable aggregate business information entity schema module is required. This schema  
728 module will contain a type definition and element declaration for every reusable ABIE in the  
729 UN/CEFACT Core Component Library. In the future this single reusable schema module may be  
730 segmented into additional modules if deemed necessary. This single reusable schema module may  
731 be compressed for runtime performance considerations if necessary. Compression means that a  
732 runtime version of the reusable ABIE schema module would be created that would consist of a  
733 subset of the ABIE constructs. This subset would consist only of those ABIEs necessary to support  
734 the specific root schema being validated.

---

735 [R31] A Reusable Aggregate Business Information Entity schema module MUST be created.

---

736 The reusable aggregate business information entity schema module will have a standardized name  
737 that uniquely differentiates it from other UN/CEFACT XSD schema modules.

---

738 [R32] The `ram:ReusableAggregateBusinessInformationEntity` schema module  
739 MUST be named 'Reusable Aggregate Business Information Entity Schema Module'.

---

#### 740 **5.5.3.5 Code List Schema Modules**

741 In cases where a code list is required or used, reusable code list schema modules will be  
742 created to minimize the impact of code list changes on root and other reusable schema. Each  
743 reusable code list schema module will contain enumeration values for codes and code values.

---

744 [R33] Reusable Code List schema modules MUST be created to convey code list  
745 enumerations.

---

746 Code list schema modules will have a standardized name that uniquely differentiates it from other  
747 UN/CEFACT XSD schema modules and external organization generated code list modules.

---

748 [R34] The name of each `clm:CodeList` schema module MUST be of the form: `<Code List`  
749 `Agency Identifier|Code List Agency Name><Code List Identification`  
750 `Identifier|Code List Name> - Code List Schema Module`

751 Where:

752 Code List Agency Identifier = Identifies the agency that maintains the code list

753 Code List Agency Name = Agency that maintains the code list

754 Code List Identification Identifier = Identifies a list of the respective corresponding codes

755 Code List Name = The name of the code list as assigned by the agency that maintains  
756 the code list

---

#### 757 **Example 5-11: Name of UN/CEFACT Account Type Code Schema Module**

```
63139 - Code List Schema Module
```

```
where:
```

```
6 = Code list agency identifier for UN/CEFACT as defined in UN/CEFACT code  
list 3055
```

```
3139 = Code list identification identifier for Contact Type Code in UN/CEFACT  
directory
```

#### 758 **Example 5-12: Name for a code using agency name and code list name**

---

```
759 Planning Level Code - Code List Schema Module
```



### 760 5.5.3.6 Identifier List Schema Modules

761 Whereas codes are normally part of a finite list that are suitable for runtime validation, identifiers may  
762 or may not be suitable for creation as a discrete list of identification schemes and subsequently  
763 validated during runtime. In those cases where runtime validation is required against a used  
764 identifier scheme, a separate identifier list schema module will be created to minimize the impact of  
765 identifier list changes on root and other reusable schema. Each reusable identifier list schema  
766 module will contain enumerated values for the identifiers.

---

767 [R35] An identifier list schema module MUST be created to convey enumerated values for  
768 each identifier list that requires runtime validation.

---

769 Identifier list schema modules will have a standardized name that uniquely differentiates it from other  
770 UN/CEFACT XSD schema modules or external organization generated schema modules.

---

771 [R36] The name of each `ids:IdentifierList` schema module MUST be of the form:  
772 `<Identifier Scheme Agency Identifier|Identifier Scheme Agency`  
773 `Name><Identifier Scheme Identifier|Identifier Scheme Name> -`  
774 `Identifier List Schema Module`

775 Where:

776 Identifier Scheme Agency Identifier = identification of the agency that maintains the  
777 identifier list

778 Identifier Scheme Agency Name = Agency that maintains the identifier list

779 Identifier Scheme Identifier = identification of the identifier list

780 Identification Scheme Name = Name as assigned by the agency that maintains the  
781 identifier list

---

#### 782 Example 5-13: Name of ISO Country Identifier schema module

```
53166-1 - Identifier List Schema Module
where:
5 = Code list agency identifier for ISO as defined in UN/CEFACT code list 3055
3166-1 = Identifier scheme identifier for Two Alpha Country Identifier in ISO
```

### 783 5.5.3.7 Other Standards Body Aggregate Business Information Entity Schema 784 Modules

785 Other Standards Body ABIE schema modules are those reusable XML constructs created by  
786 standards bodies other than UN/CEFACT and made publicly available. UN/CEFACT will only import  
787 other Standards Body ABIE schema modules when their contents are in strict conformance to the  
788 requirements of the CCTS and this specification.

---

789 [R37] Imported schema modules MUST be fully conformant with the UN/CEFACT XML  
790 *Naming and Design Rules Technical Specification* and the UN/CEFACT *Core*  
791 *Components Technical Specification*.

---

## 792 5.6 Namespace Scheme

793 A namespace is a collection of names for elements, attributes and types that serve to uniquely  
794 distinguish the collection from the collection of names in another namespace. As defined in the W3C  
795 XML specification, "XML namespaces provide a simple method for qualifying element and attribute  
796 names used in Extensible Markup Language documents by associating them with namespaces  
797 identified by URI references."<sup>5</sup> This enables interoperability and consistency in the XML artefacts for  
798 the library of reusable types and schema modules. The UN/CEFACT reusability methodology  
799 maximizes the reuse of defined named types, a combination of locally and globally declared  
800 elements, and attributes (See Section 5.4).

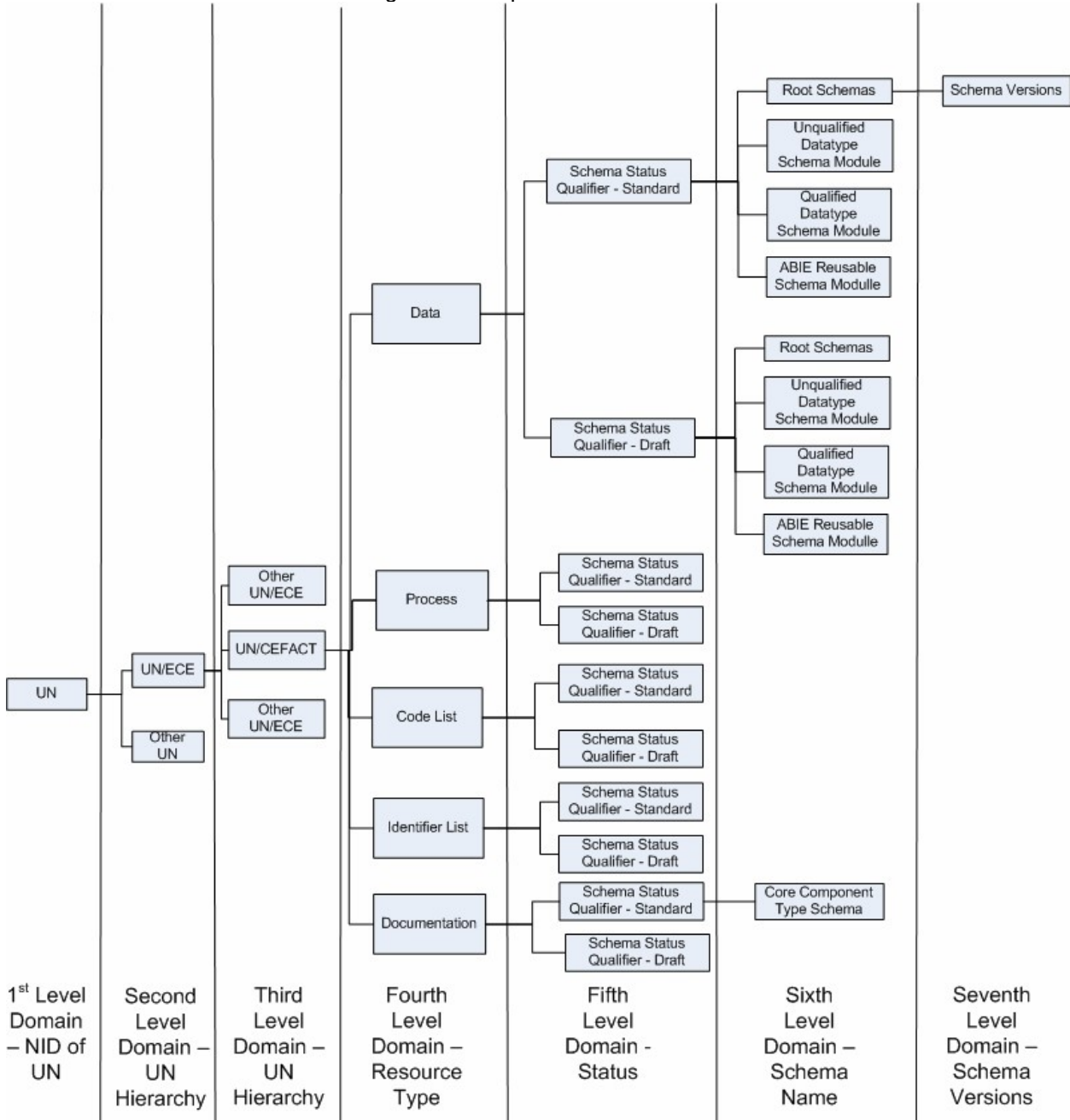
801 In addition, the modularity approach of multiple reusable schema modules (See Section 5.5)  
802 prescribe just such a method. There exist specific relationships between the various internal and  
803 external schema modules identified in Section 5.5 with respect to their namespaces. These  
804 relationships are defined in Figure 5-5. Accordingly, a sufficiently robust namespace scheme is  
805 essential.

---

<sup>5</sup> World Wide Web Consortium, *Namespaces in XML*, 14 January 1999

806 **5.6.1 Namespace Scheme**

807 In establishing a UN/CEFACT approach to namespaces, it is important to recognize that in addition  
 808 to XML requirements, many other requirements exist for a standardized namespace approach.  
 809 Accordingly, a master UN/CEFACT namespace scheme must be sufficiently flexible and robust to  
 810 accommodate both XML and other syntax requirements. Figure 5-7 reflects such an approach and  
 811 will be used as the basis for determining the namespace structure and rules that follow.



813 **Figure 5-7: UN/CEFACT Namespace Scheme**

814 **5.6.2 Declaring Namespace**

815 Best practice dictates that every schema module have its own namespace with the exception that  
 816 internal schema modules will be in the same namespace as the root schema.

817 [R38] Every UN/CEFACT defined or imported schema module MUST have a namespace  
 818 declared, using the `xsd:targetNamespace` attribute.

### 819 5.6.3 Namespace Persistence

820 Namespaces also provide a means for achieving consistency and harmonization between schema  
821 versions. UN/CEFACT has chosen to align namespace versioning with schema versioning and  
822 modularity. The UN/CEFACT modularity approach provides for grouping of reusable schemas by a  
823 root schema. Many of these schema are intended to be reused across multiple schema. Others are  
824 unique to a particular root schema. The root schema and those schema modules that are unique to  
825 it are considered a schema set. The contents of a schema set are so interrelated that proper  
826 management dictates that both versioning and namespace of all members of the set be  
827 synchronized. Schema sets are therefore assigned to a single, versioned namespace. Other  
828 schema modules are also best managed by being assigned to their own unique versioned  
829 namespaces. Accordingly, with the exception of internal schema modules, each UN/CEFACT XSD  
830 schema module will have its own namespace and each namespace will be versioned.

---

831 [R39] Every version of a defined or imported schema module other than internal schema  
832 modules MUST have its own unique namespace.

---

833 Once a namespace declaration is published, any change would result in an inability to validate  
834 instance documents citing the namespace. Accordingly, a change in the construct or contents of the  
835 namespace should not be allowed.

---

836 [R40] UN/CEFACT published namespace declarations MUST NOT be changed, and its  
837 contents MUST NOT be changed unless such change does not break backward  
838 compatibility.

---

### 839 5.6.4 Namespace Uniform Resource Identifiers

840 Namespaces must be persistent. Namespaces should be resolvable. Uniform Resource Indicators  
841 (URIs) are used for identifying a namespace. Within the URI space, options include Uniform  
842 Resource Locators (URLs) and Uniform Resource Names (URNs). URNs have an advantage in that  
843 they are persistent. URLs have an advantage in that they are resolvable. After careful  
844 consideration, UN/CEFACT has determined that URNs are most appropriate as persistence is of a  
845 higher priority, and efforts are underway to make URNs resolvable.

---

846 [R41] UN/CEFACT namespaces MUST be defined as Uniform Resource Names.

---

847 To ensure consistency, each UN/CEFACT namespace will have the same general structure. This  
848 namespace structure will follow the provisions of Internet Engineering Task Force (IETF) Request  
849 For Comments (RFC) 2141 – URN Syntax. That specification calls for a standardized URN syntax  
850 structure as follows: (phrases enclosed in quotes are REQUIRED):

851 `<URN> ::= "urn:" <NID> ":" <NSS>`

852 where :

853 `<NID>` = the Namespace Identifier

854 `<NSS>` = the Namespace Specific String.

855 The leading "urn:" sequence is case-insensitive.

856 The Namespace identifier determines the syntactic interpretation of the Namespace Specific String.  
857 Following this pattern, the UN/CEFACT namespace general structure for a namespace name should  
858 be: `urn:un:unece:unefact:<schematype>:<status>:<name>:<version>`

859 Where:

860 ○ Namespace Identifier (NID) = un

861 ○ Namespace Specific String =

862 `unece:unefact:<schematype>:<status>:<name>:<version>` with unece and  
863 unefact as fixed value second and third level domains within the NID of un

864 ○ schematype = a token identifying the type of schema module:

865 `data|process|codelist|identifierlist|documentation`

866 ○ status = the status of the schema as: `draft|standard`

867 ○ name = the name of the schema module (using upper camel case) with periods, spaces, or  
868 other separators and the words 'schema module' removed.

869 ○ version = The major version number. Sequentially assigned, first release starting with the  
870 number 1.

---

871 [R42] The names for namespaces MUST have the following structure while the schema is at  
872 draft status:  
873 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>`

874 Where:  
875 schematype = a token identifying the type of schema module:  
876 `data|process|odelist|identifierlist|documentation`  
877 status = a token identifying the standards status of the schema module:  
878 `draft|standard`  
879 name = the name of the schema module (using upper camel case) with periods, spaces,  
880 or other separators and the words 'schema module' removed.  
881 major = the major version number. Sequentially assigned, first release starting with the  
882 number 1.  
883

884 [R43] This rule was combined with [R42].

---

#### 885 **Example 5-14: Namespace Name at Draft Status**

886 `"urn:un:unece:uncefact:data:draft:UnqualifiedDataType:1"`

#### 887 **Example 5-15: Namespace Name at Specification Status**

888 `"urn:un:unece:uncefact:data:standard:UnqualifiedDataType:1"`

### 889 **5.6.5 Namespace Constraint**

890 To ensure consistency in declaring namespaces, a namespace should only be declared for an XML  
891 construct by the owner of that namespace – unless specifically designed as a generic namespace  
892 such as xsi. Accordingly, UN/CEFACT namespaces will only contain XML constructs created and  
893 assigned by UN/CEFACT.

---

894 [R44] UN/CEFACT namespace values will only be assigned to UN/CEFACT developed  
895 objects.

---

### 896 **5.6.6 UN/CEFACT XSD Namespace Schema Tokens**

897 Namespace URIs are typically represented by tokens rather than citing the entire URI as the  
898 qualifier in qualified XML constructs. UN/CEFACT has developed a token pattern for each type of  
899 UN/CEFACT schema module. These token patterns are identified in the applicable schema  
900 module subsection in Section 7.

### 901 **5.7 Schema Location**

902 Schema locations are required to be in the form of a URI scheme. Schema locations are typically  
903 based on their namespaces. Schema locations are typically defined as URL based URI schemes  
904 because of resolvability limitations of URN based URI scheme. However, UN/CEFACT XSD  
905 Schema use a URN based URI scheme for namespace declarations because persistence is  
906 considered more important than resolvability. In recognition of the need for resolvability of schema  
907 location, until such time as URNs become fully resolvable, UN/CEFACT will store schema in  
908 locations identified using a URL based URI scheme.

---

909 [R45] The general structure for schema location MUST be:  
910 `../<schematype>/<status>/<name>_<major>.<minor>[p <revision>].xsd`

911 Where:  
912 schematype = a token identifying the type of schema module:  
913 `data|process|odelist|identifierlist|documentation`  
914 status = the status of the schema as: `draft|standard`  
915 name = the name of the schema module (using upper camel case) with periods,  
916 spaces, or other separators and the words 'schema module' removed.  
917 major = the major version number, sequentially assigned, first release starting with the  
918 number 1.  
919 minor = the minor version number within a major release, sequentially assigned, first  
920 release starting with the number 0.

- 921 revision = sequentially assigned alphanumeric character for each revision of a minor  
922 release. Only applicable where status = draft.
- 923 [R46] Each `xsd:schemaLocation` attribute declaration MUST contain a resolvable URL, and  
924 in the case of an absolute path, a persistent URL.
- 925 [R47] This rule has been removed.
- 

## 926 5.8 Versioning

927 The versioning scheme for UN/CEFACT XSD schema modules is composed of a major version  
928 number and where appropriate, a minor version number. Major version numbers are reflected in the  
929 namespace declaration while minor version numbers are only reflected in the schema location.  
930 Major and minor version numbers are also declared in the version attribute in the `xsd:schema`  
931 element.

- 932 [R48] The `xsd:schema` version attribute MUST always be declared.
- 933 [R49] The `xsd:schema` version attribute MUST use the following template:  
934 `<xsd:schema ... version="<major>.<minor>">`
- 935 [R50] Every schema version namespace declaration MUST have the URI of:  
936 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>`
- 

### 937 5.8.1 Major Versions

938 A major version of a UN/CEFACT XSD schema module constitutes significant and/or non-backwards  
939 compatible changes. If any XML instance based on such older major version UN/CEFACT XSD  
940 Schema attempts validation against the newer version, it may experience validation errors. A new  
941 major version will be produced when significant and/or non-backward compatible changes occur, i.e.

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

947 Major version numbers are reflected in the namespace declaration as follows:

948 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>` Where:

- 949 ○ major = the first version starts with the number 1.

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

- 953 [R51] Every UN/CEFACT XSD Schema and schema module major version number MUST be  
954 a sequentially assigned incremental integer greater than zero.
- 

### 955 5.8.2 Minor Versions

956 Within a major version of an UN/CEFACT XSD schema module there can be a series of minor, or  
957 backward compatible, changes. The minor versioning of an UN/CEFACT XSD schema module  
958 determines its compatibility with UN/CEFACT XSD schema modules with preceding and subsequent  
959 minor versions within the same major version. The minor versioning scheme thus helps to establish  
960 backward and forward compatibility. Minor versions will only be increased when compatible changes  
961 occur, i.e.

- 962 ○ Adding values to enumerations
- 963 ○ Optional extensions
- 964 ○ Add optional elements

- 965 [R52] Minor versioning MUST be limited to declaring new optional XSD constructs, extending  
966 existing XSD constructs, or refinements of an optional nature.
-

967 Minor versions are reflected in the schema location as identified in section 5.7, but are not reflected  
968 in the namespace declaration. Minor versions will be declared using the `xsd:version` attribute in  
969 the `xsd:schema` element. It is only necessary to declare the minor version in the internal schema  
970 version attribute since instance documents with different minor versions are compatible with the  
971 major version held in the same namespace. By using the version attribute in each document  
972 instance, the application can provide the appropriate logic switch for different compatible versions  
973 without having knowledge of the schema version at which the document instance was delivered.

974 Just like major version numbers, minor version numbers should be based on logical progressions to  
975 ensure semantic understanding of the approach and guarantee consistency in representation. Non-  
976 negative, sequentially assigned incremental integers satisfy this requirement.

977 Minor version changes are not allowed to break compatibility with previous minor versions.  
978 Compatibility includes consistency in naming of the schema constructs to include elements,  
979 attributes, and types. UN/CEFACT minor version changes will not include renaming the schema  
980 construct.

---

981 [R53] For UN/CEFACT minor version changes, the name of the schema construct MUST NOT  
982 change.

---

983 Semantic compatibility across minor versions is essential.

---

984 [R54] Changes in minor versions MUST NOT break semantic compatibility with prior versions  
985 having the same major version number.

---

986 For a particular namespace, the parent major version and subsequent minor versions of a major  
987 version establish a linearly linked relationship. Since each major version is assigned its own  
988 namespace, for conformance purposes, the first minor version must incorporate all XML constructs  
989 present in the parent major version, and each new minor version needs to incorporate all XML  
990 constructs present in the immediately preceding minor version.

---

991 [R55] UN/CEFACT minor version schema MUST incorporate all XML constructs from the  
992 immediately preceding major or minor version schema.

---



## 993 6 General XML Schema Language Conventions

### 994 6.1 Schema Construct

- 
- 995 [R56] The `xsd:elementFormDefault` attribute MUST be declared and its value set to  
996 `qualified`.
- 997 [R57] The `xsd:attributeFormDefault` attribute MUST be declared and its value set to  
998 `unqualified`.
- 999 [R58] The `xsd` prefix MUST be used in all cases when referring to  
1000 <http://www.w3.org/2001/XMLSchema> as follows:  
1001 `xmlns:xsd=http://www.w3.org/2001/XMLSchema`.
- 

#### 1002 Example 6-1: Element and Attribute Form Default

```
1003 <xsd:schema targetNamespace=" ... see namespace ...  
1004 xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
1005 elementFormDefault="qualified"  
1006 attributeFormDefault="unqualified">
```

### 1007 6.1.1 Constraints on Schema Construction

- 
- 1008 [R59] `xsd:appInfo` MUST NOT be used.
- 1009 [R60] `xsd:notation` MUST NOT be used.
- 1010 [R61] `xsd:wildcard` MUST NOT be used.
- 1011 [R62] The `xsd:any` element MUST NOT be used.
- 1012 [R63] The `xsd:any` attribute MUST NOT be used.
- 1013 [R64] Mixed content MUST NOT be used (excluding documentation).
- 1014 [R65] `xsd:substitutionGroup` MUST NOT be used.
- 1015 [R66] `xsd:ID/xsd:IDREF` MUST NOT be used.
- 1016 [R67] `xsd:key/xsd:keyref` MUST be used for information association.
- 1017 [R68] The absence of a construct or data MUST NOT carry meaning.
- 

## 1018 6.2 Attribute and Element Declarations

### 1019 6.2.1 Attributes

#### 1020 6.2.1.1 Usage of Attributes

1021 User declared attributes are only used to convey the supplementary components of core component  
1022 types. However, predefined `xsd:attributes` will be used as described elsewhere in this  
1023 document.

- 
- 1024 [R69] User declared attributes MUST only be used to convey core component type (CCT)  
1025 supplementary component information.
- 

1026 The user declared attributes can represent different types of values. Some of the values can be  
1027 variable information or can be based on code lists or identifier schemes.

- 
- 1028 [R70] A `xsd:attribute` that represents a supplementary component with variable  
1029 information MUST be based on the appropriate XSD built-in data type.
- 1030 [R71] A `xsd:attribute` that represents a supplementary component which represents  
1031 codes MUST be based on the `xsd:simpleType` of the appropriate code list.
-

---

1032 [R72] A **xsd:attribute** that represents a supplementary component which represents  
1033 identifiers MUST be based on the **xsd:simpleType** of the appropriate identifier  
1034 scheme.

---

### 1035 **6.2.1.2 Constraints on Attribute Declarations**

1036 In general, the absence of an element in an XML schema does not have any particular meaning - it  
1037 may indicate that the information is unknown, or not applicable, or the element may be absent for  
1038 some other reason. The XML schema specification does however provide a feature, the  
1039 **xsd:nillable** attribute, whereby an element may be transferred with no content, but still use its  
1040 attributes and thus carry semantic meaning. In order to respect the principles of the CCTS and to  
1041 retain semantic clarity the nillability feature of XSD will not be used.

---

1042 [R73] The **xsd:nillable** attribute MUST NOT be used.

---

## 1043 **6.2.2 Elements**

### 1044 **6.2.2.1 Usage of Elements**

1045 Elements are declared for the document level business information payload, ABIEs, BBIEs, and  
1046 ASBIEs.

### 1047 **6.2.2.2 Element Declaration**

---

1048 [R74] Empty elements MUST NOT be used.

1049 [R75] Every BBIE leaf element declaration MUST be of the **udt:UnqualifiedDataType** or  
1050 **qdt:QualifiedDataType** that represents the source basic business information  
1051 entity (BBIE) data type.

---

### 1052 **Example 6-2: Element Declaration**

```
1053 <xsd:complexType name="AcknowledgementType">  
1054 <xsd:annotation>  
1055 ... see annotation ...  
1056 </xsd:annotation>  
1057 <xsd:sequence>  
1058 <xsd:element name="AcknowledgementDocument"  
1059 type="ram:AcknowledgementDocumentType" minOccurs="0">  
1060 <xsd:annotation>  
1061 ... see annotation ...  
1062 </xsd:annotation>  
1063 </xsd:element>  
1064 <xsd:element name="ProjectParty" type="ram:ProjectPartyType"  
1065 <xsd:annotation>  
1066 ... see annotation ...  
1067 </xsd:annotation>  
1068 </xsd:element>  
1069 </xsd:sequence>  
1070 </xsd:complexType>
```

### 1074 **6.2.2.3 Constraints on Element Declarations**

---

1075 [R76] The **xsd:all** element MUST NOT be used.

---

## 1076 **6.3 Type Declarations**

### 1077 **6.3.1 Usage of Types**

---

1078 [R77] All type definitions MUST be named.

---

1079



1080

**Example 6-3: Type Definition Name**1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088

```

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

```

1089  
1090  
1091

Data types are intended to be reused to the maximum extent possible. If an existing data type has the same semantic meaning and structure (facet restrictions) as the intended data type, then the existing data type should be used rather than creating a semantically equivalent duplicate data type.

1092  
1093

---

[R78] Data type definitions with the same semantic meaning MUST NOT have an identical set of facet restrictions.

---

1094

## 6.3.2 Simple Type Definitions

1095  
1096  
1097

**xsd:simpleTypes** must always be used where they satisfy the user's business requirements. Where these business requirements cannot be satisfied, user defined complex type definitions will be used.

1098

**Example 6-4: Simple Types in Unqualified Data Type Schema Module**1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106

```

<xsd:simpleType name="TextType">
  <xsd:annotation>
    ... see
    annotation
    ...
  </xsd:annotation>
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

```

1107

**Example 6-5: Simple Types in Code Lists Module**1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117

```

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

```

1118

## 6.3.3 Complex Type Definitions

1119  
1120

User defined complex types may be used when XSD built-in data types do not satisfy the business requirements or when an aggregate business information entity (ABIE) must be defined.

1121

**Example 6-6: Complex Type of Object Class "ProjectContactType"**1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129

```

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

```

1130

## 6.4 User of XSD Extension and Restriction

1131  
1132  
1133  
1134  
1135  
1136

The general philosophy is that all UN/CEFACT XSD schema constructs will follow the model defined in Figure 5.1. These schema constructs are based on the concept that the underlying semantic structures of the core components and business information entities are normative forms of standards that developers are not allowed to alter without coordination of appropriate UN/CEFACT Domains. Accordingly, as business requirements dictate, new schema constructs will be created and new types defined and elements declared as appropriate. The concept of derivation through the

1137 use of `xsd:extension` and `xsd:restriction` will only be used in limited circumstances as  
1138 described below.

## 1139 6.4.1 Extension

---

1140 [R79] `xsd:extension` MUST only be used in the `cct:CoreComponentType` schema  
1141 module and the `udt:UnqualifiedDataType` schema module. When used it MUST  
1142 only be used for declaring `xsd:attributes` to accommodate relevant supplementary  
1143 components.

---

## 1144 6.4.2 Restriction

1145 The CCTS specification employs the concept of semantic restriction in creating specific instantiations  
1146 of core components. Accordingly, `xsd:restriction` will be used as appropriate to define types  
1147 that are derived from the existing types. Where used, the derived types must always be renamed.  
1148 Simple and complex type restrictions may be used. `xsd:restriction` can be used for facet  
1149 restriction and/or attribute restriction.

---

1150 [R80] When `xsd:restriction` is applied to a `xsd:simpleType` or `xsd:complexType`  
1151 that represents a data type the derived construct MUST use a different name.

---

### 1152 Example 6-7: Restriction of Simple Type

```
1153 <xsd:simpleType name="TaxAmountType">  
1154   <xsd:annotation>  
1155     ... see  
1156     annotation ...  
1157   </xsd:annotation>  
1158   <xsd:restriction base="udt:AmountType">  
1159     <xsd:totalDigits value="10"/>  
1160     <xsd:fractionDigits value="3"/>  
1161   </xsd:restriction>  
1162 </xsd:simpleType>
```

## 1163 6.5 Annotation

1164 All UN/CEFACT XSD schema constructs will use `xsd:annotation` to provide the documentation  
1165 specified in Section 7 of CCTS.

---

1166 [R81] Each UN/CEFACT defined or declared construct MUST use the `xsd:annotation`  
1167 element for required CCTS documentation.

---

### 1168 [Note]

1169 In order to conform to this specification, this rule also applies to any construct imported  
1170 from other standards bodies.

## 1171 6.5.1 Documentation

1172 The annotation documentation will be used to convey all metadata as specified in the CCTS, i.e., to  
1173 convey the semantic content carried in the XML construct. Therefore, all elements specified for the  
1174 documentation are defined in the Core Component Technical Specification namespace. The  
1175 current version of this namespace is:  
1176 `urn:un:unece:uncefact:documentation:standard:CoreComponentsTechnicalSpeci`  
1177 `fication:2.`

1178 Thus, all schema modules must contain the following namespace declaration:  
1179 `ccts="urn:un:unece:uncefact:documentation:standard:CoreComponentsTechnic`  
1180 `alSpecification:2."`

1181 and all documentation elements must be prefixed with 'ccts'.

1182 The following annotations are required as defined in section 7 in type definitions and element  
1183 declarations (the representation of each item in XML code is shown in parenthesis):

- 1184 ○ **Unique Identifier:** The unique identifier assigned to the artefact in the library. (UniqueID)

- 1185 ○ **Acronym:** The abbreviation of the type of component.
- 1186 (Acronym)
- 1187     ▪ **BBIE** – Basic Business Information Entity
- 1188     ▪ **ABIE** – Aggregate Business Information Entity
- 1189     ▪ **ASBIE** – Associated Business Information Entity
- 1190     ▪ **CCT** – Core Component Type
- 1191     ▪ **QDT** – Qualified Data Type
- 1192     ▪ **UDT** – Unqualified Data Type
- 1193 ○ **Dictionary Entry Name:** The complete name (not the tag name) of the artefact in the
- 1194 library. (DictionaryEntryName)
- 1195 ○ **Name:** The name of the supplementary component or business information payload.
- 1196 (Name)
- 1197 ○ **Version:** The version of the artefact as assigned by the registry. (Version)
- 1198 ○ **Definition:** The semantic meaning of the artefact. (Definition)
- 1199 ○ **Cardinality:** An indication of whether the property represents a not-applicable,
- 1200 optional, mandatory and/or repetitive characteristic of the object. (Cardinality)
- 1201 ○ **Object Class Term:** The Object Class represented by the artefact. (ObjectClassTerm)
- 1202 ○ **Object Class Qualifier Term:** A term(s) that qualifies the Object
- 1203 Class. (ObjectClassQualifierTerm)
- 1204 ○ **Property Term:** The Property Term represented by the artefact. (PropertyTerm)
- 1205 ○ **Property Qualifier Term:** A term(s) that qualifies the Property Term.
- 1206 (PropertyQualifierTerm)
- 1207 ○ **Associated Object Class Term:** The Associated Object Class Term represented by the
- 1208 artefact. (AssociatedObjectClassTerm)
- 1209 ○ **Associated Object Class Qualifier Term:** A term(s) that qualifies the Associated Object
- 1210 ClassTerm. (AssociatedObjectClassQualifierTerm)
- 1211 ○ **Association Type:** The association type of the Association Business Information
- 1212 Entity. (AssociationType)
- 1213 ○ **Primary Representation Term:** The Primary Representation Term represented by the
- 1214 artefact. (PrimaryRepresentationTerm)
- 1215 ○ **Data Type Qualifier Term:** A term(s) that qualifies the Data Type
- 1216 Term. (DataTypeQualifierTerm)
- 1217 ○ **Primitive Type:** The primitive data type as assigned to the artefact by CCTS.
- 1218 (PrimitiveType)
- 1219 ○ **Business Process Context Value:** A valid value describing the Business Process
- 1220 contexts for which this construct has been designed. Default is 'In All Contexts'.
- 1221 (BusinessProcessContextValue)
- 1222 ○ **Geopolitical/Region Context Value:** A valid value describing the Geopolitical/Region
- 1223 contexts for which this construct has been designed. Default is 'In All Contexts'.
- 1224 (GeopoliticalOrRegionContextValue)
- 1225 ○ **Official Constraints Context Value:** A valid value describing the Official Constraints
- 1226 contexts for which this construct has been designed. Default is 'None'.
- 1227 (OfficialConstraintContextValue)
- 1228 ○ **Product Context Value:** A valid value describing the Product contexts for which this
- 1229 construct has been designed. Default is 'In All Contexts'. (ProductContextValue)
- 1230 ○ **Industry Context Value:** A valid value describing the Industry contexts for which this
- 1231 construct has been designed. Default is 'In All Contexts'. (IndustryContextValue)
- 1232 ○ **Business Process Role Context Value:** A valid value describing the Role contexts for
- 1233 which this construct has been designed. Default is 'In All Contexts'.
- 1234 (BusinessProcessRoleContextValue)
- 1235 ○ **Supporting Role Context Value:** A valid value describing the Supporting Role contexts for
- 1236 which this construct has been designed. Default is 'In All Contexts'.
- 1237 (SupportingRoleContextValue)
- 1238 ○ **System Capabilities Context Value:** A valid value describing the Systems Capabilities
- 1239 contexts for which this construct has been designed. Default is 'In All Contexts'.
- 1240 (SystemCapabilitiesContextValue)

- 1241 ○ **Usage Rule:** A constraint that describes specific conditions which are applicable to the
- 1242 artefact. (UsageRule)
- 1243 ○ **Business Term:** A synonym term under which the artefact is commonly known and
- 1244 used in business. (BusinessTerm)
- 1245 ○ **Example:** A possible value for the artefact. (Example)

1246 Appendix F specifies normative information on the specific annotation required for each of the  
 1247 artefacts.

1248 Note: The list above defines the minimum annotation documentation requirements.  
 1249 However, additional annotation documentation may be included when necessary.

1250 **Example 6-8: Example of annotation**

```

1251 <xsd:annotation>
1252   <xsd:documentation xml:lang="en">
1253     <ccts:UniqueID>UN01005559</ccts:UniqueID>
1254     <ccts:Acronym>BBIE</ccts:Acronym>
1255     <ccts:DictionaryEntryName>CI Note. Content. Code</ccts:DictionaryEntryName>
1256     <ccts:Version>1.0</ccts:Version>
1257     <ccts:Definition>The code specifying the content of this CI
1258     note.</ccts:Definition>
1259     <ccts:Cardinality>0..1</ccts:Cardinality>
1260     <ccts:ObjectClassTerm>Note</ccts:ObjectClassTerm>
1261     <ccts:ObjectClassQualifierTerm>CI</ccts:ObjectClassQualifierTerm>
1262     <ccts:PropertyTerm>Content</ccts:PropertyTerm>
1263     <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>
1264     <ccts:BusinessProcessContextValue>Cross Industry
1265     Trade</ccts:BusinessProcessContextValue>
1266     <ccts:GeopoliticalOrRegionContextValue>In All
1267     Contexts</ccts:GeopoliticalOrRegionContextValue>
1268     <ccts:OfficialConstraintContextValue>None</ccts:OfficialConstraintContextValue>
1269     <ccts:ProductContextValue>In All Contexts</ccts:ProductContextValue>
1270     <ccts:IndustryContextValue>In All Contexts</ccts:IndustryContextValue>
1271     <ccts:BusinessProcessRoleContextValue>In All
1272     Contexts</ccts:BusinessProcessRoleContextValue>
1273     <ccts:SupportingRoleContextValue>In All
1274     Contexts</ccts:SupportingRoleContextValue>
1275     <ccts:SystemCapabilitiesContextValue>In All
1276     Contexts</ccts:SystemCapabilitiesContextValue>
1277   </xsd:documentation>
1278 </xsd:annotation>
  
```

1279 Each UN/CEFACT construct containing a code should include documentation that will identify the  
 1280 code list(s) that must be minimally supported when the construct is used.

1281 The following table provides a summary view of the annotation data as defined in section 6.

	rem:RootSchema	ABIE: xsd:complexType and xsd:element	BBIE: xsd:element	ASBIE: xsd:element	cct:CoreComponentType	Supplementary component	udt:UnqualifiedData Type	qdt:QualifiedData Type
Unique Identifier	M	M	M	M	M	O	M	M
Acronym	M	M	M	M	M	M	M	M
Dictionary Entry Name		M	M	M	M	M	M	M
Name	M							
Version	M	M	M	M	M		M	M
Definition	M	M	M	M	M	M	M	M
Cardinality			M	M		M		
Object Class Term		M	M	M		M		
Object Class Qualifier Term		O	O	O				
Property Term			M	M		M		
Property Qualifier Term			O	O				
Associated Object Class Term				M				
Associated Object Class Qualifier Term				O				
Association Type				M				
Primary Representation Term			M		M	M	M	M
Data Type Qualifier Term								M
Primitive Type					M	M	M	M
Business Process Context Value	M, R	O, R	O, R	O, R				O, R
Geopolitical/Region Context Value	O, R	O, R	O, R	O, R				O, R
Official Constraints Context Value	O, R	O, R	O, R	O, R				O, R

	rem:RootSchema	ABIE: xsd:complexType and xsd:element	BBIE: xsd:element	ASBIE xsd:element	cct:CoreComponentType	Supplementary component	udt: UnqualifiedDataType	qdt: QualifiedData Type
Product Context Value	O, R	O, R	O, R	O, R				O, R
Industry Context Value	O, R	O, R	O, R	O, R				O, R
Business Process Role Context Value	O, R	O, R	O, R	O, R				O, R
Supporting Role Context Value	O, R	O, R	O, R	O, R				O, R
System Capabilities Context Value	O, R	O, R	O, R	O, R				O, R
Usage Rule		O, R	O, R	O, R	O, R	O, R	O, R	O, R
Business Term		O, R	O, R	O, R				
Example		O, R	O, R	O, R				O, R

1282

1283

Key:

1284

M - mandatory

1285

O - optional

1286

R - repeating

1287

Note

1288

When a particular optional annotation element contains no value, it may be omitted from the

1289

schema.

## 1290 7 XML Schema Modules

1291 This section describes the requirements of the various XML schema modules that will be incorporated within  
1292 the UN/CEFACT library.

### 1293 7.1 Root Schema

1294 The root schema serves as the container for all other schema content that is required to fulfil a business  
1295 information exchange. The root schema resides in its own namespace and imports external schema modules  
1296 as needed. It may also include internal schema modules that reside in its namespace.

#### 1297 7.1.1 Schema Construct

1298 Each root schema will be constructed in a standardized format in order to ensure consistency and ease of  
1299 use. The specific format is shown in the example below and must adhere to the format of the relevant  
1300 sections as detailed in Appendix B.

#### 1301 Example 7-1: Structure of RootSchema Module

```
1302 <?xml version="1.0" encoding="UTF-8"?>
1303 <!-- ===== -->
1304 <!-- ===== [MODULENAME] Schema Module ===== -->
1305 <!-- ===== -->
1306 <!--
1307 Schema agency: UN/CEFACT
1308 Schema version: 2.0
1309 Schema date: [SCHEMADATE]

... see intellectual property disclaimer ...
-->
<xsd:schema
targetNamespace="urn:un:unece:uncefact:data:draft:[MODULENAME]:1"
... see namespaces ...
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
<!-- ===== -->
<!-- ===== Imports ===== -->
<!-- ===== -->
<!-- ===== Import of [MODULENAME] ===== -->
<!-- ===== -->
< see imports
<!-- ===== -->
<!-- ===== Include ===== -->
<!-- ===== -->
<!-- ===== Include of [MODULENAME] ===== -->
<!-- ===== -->
... see includes ...
<!-- ===== -->
<!-- ===== Element Declarations ===== -->
<!-- ===== -->
<!-- ===== Root Element Declarations ===== -->
<!-- ===== -->
See element declarations...
<!-- ===== -->
<!-- ===== Type Definitions ===== -->
<!-- ===== -->
<!-- ===== Type Definitions: [TYPE] ===== -->
<!-- ===== -->
<xsd:complexType name="[TYPENAME]">
<xsd:restriction base="xsd:token">
... see type definition ...
</xsd:restriction>
</xsd:complexType>
</xsd:schema>
```

1346 **7.1.2 Namespace Scheme**

1347 All root schemas published by UN/CEFACT will be assigned a unique token by BPS to represent the  
1348 namespace prefix. This token will be prefixed by 'rsm'.

1349 [R82] The root schema module MUST be represented by a unique token.

1350 **Example 7-2: Namespace of Root Schema Module**

1351 `"xmlns:rsm="urn:un:unece:uncefact:data:draft:CrossIndustryInvoice:1"`

1352 [Note]

1353 Throughout this specification, the token 'rsm' is used for the unique root schema token.

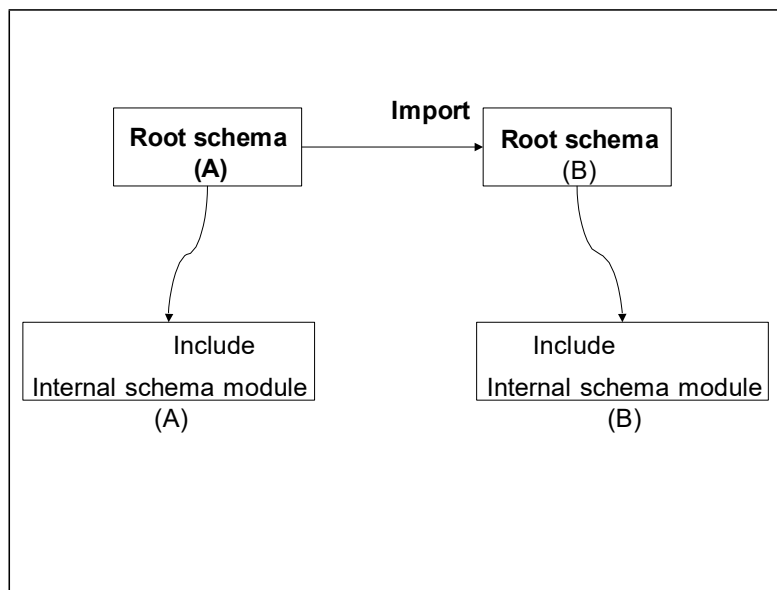
1354 **7.1.3 Imports and Includes**

1355 [R83] The `rsm:RootSchema` MUST import the following schema modules:

- 1356 – `ram:ReusableABIE` Schema Module
- 1357 – `udt:UnqualifiedDataType` Schema Module
- 1358 – `qdt:QualifiedDataType` Schema Module

1359 The root schema will include all internal schema modules that reside in its namespace. The root schema may  
1360 import other external schema modules as necessary provided they conform to UN/CEFACT naming and  
1361 design rules. One root schema (root schema A) may also make use of ABIEs defined as part of another root  
1362 schema (root schema B) or that root schema's internal schema module. In other words, reuse type  
1363 definitions and element declarations defined in another namespace. An example may be that the root schema  
1364 for a Purchase Order Response message (root schema A) makes use of ABIEs defined as part of the schema  
1365 definition for a Purchase Order Request message (root schema B). If that is the case then such type  
1366 definitions and element declarations should be imported in to the root schema (root schema A). To achieve  
1367 this only the root schema (root schema B) in the namespace containing the type definitions and element  
1368 declarations needed should be imported as this in itself included the subordinate internal schema modules.

1369



1276 **Figure 7-1: Imports and Includes of Schema Modules in Root Schema**

- 1277 [R84] A `rsm:RootSchema` in one UN/CEFACT namespace that is dependent upon type definitions or  
1278 element declaration defined in another namespace MUST import the `rsm:RootSchema` from  
1279 that namespace.
- 1280 [R85] A `rsm:RootSchema` in one UN/CEFACT namespace that is dependent upon type definitions or  
1281 element declarations defined in another namespace MUST NOT import Schema Modules from  
1282 that namespace other than the `rsm:RootSchema`.
- 1283 [R86] The `rsm:RootSchema` MUST include any internal schema modules that reside in the root  
1284 schema namespace.

1285 **7.1.4 Root Element Declaration**

1286 Each UN/CEFACT business information payload message has a single root element that is globally declared  
1287 in the root schema. The global element is named according to the business information payload that it  
1288 represents and references the target information payload that contains the actual business information.<sup>6</sup>

- 1289 [R87] A single global element known as the root element, representing the business information  
1290 payload, MUST be declared in a `rsm:RootSchema`.
- 1291 [R88] The name of the root element MUST be the name of the business information payload with  
1292 separators and spaces removed.
- 1293 [R89] The root element declaration must be of `xsd:complexType` that represents the business  
1294 information payload.

1295 **Example 7-3: Name of Root Element**

```
1296 <!-- =====>  
1297 <!-- ===== Root Element ===== -->  
1298 <!-- =====>  
1299 <xsd:element name="CrossIndustryInvoice" type="rsm:CrossIndustryInvoiceType">  
1300 <xsd:annotation>  
1301 ... see annotation ...  
1302 </xsd:annotation>  
1303 </xsd:element>
```

1304 **7.1.5 Type Definitions**

1305 Root schemas are limited to defining a single `xsd:complexType` and a declaring a single global element  
1306 that fully describe the business information payload.

- 1307 [R90] Root schema MUST define a single `xsd:complexType` that fully describes the business  
1308 information payload.
- 1309 [R91] The name of the root schema `xsd:complexType` MUST be the name of the root element with  
1310 the word 'Type' appended.

1311 **Example 7-4: Name of Complex Type Definition**

```
1312 <!-- =====>  
1313 <!-- ===== Root Element ===== -->  
1314 <!-- =====>  
1315 <xsd:element name="PurchaseOrderRequest" type="rsm:PurchaseOrderRequestType">  
1316 <xsd:annotation>  
1317 ... see annotation ...  
1318 </xsd:annotation>  
1319 </xsd:element>  
1320 <xsd:complexType name="PurchaseOrderRequestType">  
1321 <xsd:sequence>  
1322 ...  
1323 </xsd:sequence>
```

<sup>6</sup> All references to root element represent the globally declared element in a UN/CEFACT schema module that is designated as the root element for instances that use that schema.



1324

</xsd:complexType>

1325

## 7.1.6 Annotations

1326

[R92] The `rsm:RootSchema` root element declaration MAY have a structured set of annotations present in the following pattern:

1327

1328

○ UniqueID (required): The identifier that references the business information payload instance in a unique and unambiguous way.

1329

○ Acronym (required): The abbreviation of the type of component. In this case the value will always be RSM.

1330

○ Name (required): The name of the business information payload.

1331

○ Version (required): An indication of the evolution over time of a business information payload.

1332

○ Definition (required): A brief description of the business information payload.

1333

○ BusinessProcessContextValue (required, repetitive): The business process with which this business information is associated.

1334

○ GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for this business information payload.

1335

○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this business information payload.

1336

○ ProductContextValue (optional, repetitive): The product context for this business information payload.

1337

○ IndustryContextValue (optional, repetitive): The industry context for this business information payload.

1338

○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this business information payload.

1339

○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this business information payload.

1340

○ SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for this business information payload.

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

## 7.2 Internal Schema

1353

A UN/CEFACT internal schema module will contain schema constructs representing ABIEs that are

1354

specific to a given root schema, such as restricted ABIEs created through CCBDA. Internal schema

1355

modules reside in the same namespace as their root schema. These constructs are subject to the same

1356

rules as those for reusable ABIEs as provided in sections 7.3.4, 7.3.5, and 7.3.6.

1357

### 7.2.1 Schema Construct

1358

Each internal schema will be constructed in a standardized format in order to ensure consistency and ease of

1359

use. Each internal schema format must adhere to the format of the relevant sections as detailed in Appendix

1360

B.

1361

### 7.2.2 Namespace Scheme

1362

[R93] All UN/CEFACT internal schema modules MUST be in the same namespace as their corresponding `rsm:RootSchema`.

1363

1364

The UN/CEFACT internal schema modules do not declare a target namespace, but instead reside in the

1365

namespace of their parent root schema. All internal schema modules are accessed from the root schema

1366

using `xsd:include`.

1367

[R94] The internal schema module MUST be represented by the same token as its `rsm:RootSchema`.

1368

### 7.2.3 Imports and Includes

1369

The internal schema module may import or include other schema module as necessary to support

1370

validation.

1371

## 7.3 Reusable Aggregate Business Information Entity Schema

1372

The UN/CEFACT ABIE schema module is a schema instance that contains all of the reusable ABIEs. This schema module may thus be used (imported into) in conjunction with any of the UN/CEFACT root schema.

1373

1374

### 7.3.1 Schema Construct

1375

The reusable ABIE schema will be constructed in a standardized format in order to ensure consistency and ease of use. The specific format is shown below and must adhere to the format of the relevant sections as detailed in Appendix B.

1376

1377

1378

#### Example 7-5: Structure of Reusable ABIEs Schema Module

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- =====>
<!-- ===== Reusable ABIEs Schema Module ===== -->
<!-- =====>
<!--
Schema agency:    UN/CEFACT
Schema version:   2.0
Schema date:      [SCHEMADATE]

... see intellectual property disclaimer ...

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

1401

### 7.3.2 Namespace Scheme

1402

[R95] The Reusable Aggregate Business Information Entity schema module MUST be represented by the token **ram**.

1403

1404

#### Example 7-6: Namespace of Reusable Aggregate Business Information Entity Schema Module

1405

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

1406

#### Example 7-7: Schema-Element of Reusable ABIEs Schema Module

1407

1408

1409

1410

```
<xsd:schema targetNamespace=
"urn:un:unece:uncefact:data:draft:ReusableAggregateBusinessInformationEntity:1"
xmlns:ram=
"urn:un:unece:uncefact:data:draft:ReusableAggregateBusinessInformation:1"
```

1411

### 7.3.3 Imports and Includes

1412

[R96] The **ram:ReusableAggregateBusinessInformationEntity** schema MUST import the following schema modules:

1413

1414

– **udt:UnqualifiedDataType** Schema Module

1415

– **qdt:QualifiedDataType** Schema Module

1416

#### Example 7-8: Import of required modules

1417

1418

1419

1420

1421

1422

1423

1424

```
<!-- =====>
<!-- ===== Imports ===== -->
<!-- =====>
<!-- ===== Import of Qualified Data Type Schema Module ===== -->
<!-- =====>
<xsd:import namespace=
"urn:un:unece:uncefact:data:draft:QualifiedDataType:1"
schemaLocation="http://www.unece.org/uncefact/data/draft/QualifiedDataType 1.xsd"/
```

1425  
1426  
1427  
1428  
1429  
1430  
1431

```
>  
<!-- ===== -->  
<!-- ===== Import of Unqualified Data Type Schema Module ===== -->  
<!-- ===== -->  
<xsd:import  
  namespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataType:1"  
  schemaLocation="http://www.unece.org/uncefact/data/draft/UnqualifiedDataTypes_1.xsd"/>
```

1432

### 7.3.4 Type Declarations

1433  
1434

[R97] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named **xsd:complexType** MUST be defined.

1435  
1436  
1437

[R98] The name of the ABIE **xsd:complexType** MUST be the **ccts:DictionaryEntryName** with the spaces and separators removed, approved abbreviations and acronyms applied, and with the 'Details' suffix replaced with 'Type'.

1438  
1439  
1440  
1441

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

1442  
1443  
1444

[R99] Every aggregate business information entity (ABIE) **xsd:complexType** definition content model MUST use the **xsd:sequence** and/or **xsd:choice** elements to reflect each property (BBIE or ASBIE) of its class.

1445

[R100] Recursion of **xsd:sequence** and/or **xsd:choice** MUST NOT occur.

1446  
1447  
1448  
1449  
1450

No complex type may contain a sequence followed by another sequence or a choice followed by another choice. However, it is permissible to alternate sequence and choice as in example 7.9. Note that the choice construction will not be used in the base reusable ABIE UN/CEFACT schemas, as it cannot be directly modeled in CCTS. However, third party schemas that implement those restrictions would still be conformant.

1451

#### Example 7-9: Sequence within an object class

1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476

```
<xsd:complexType name="AcknowledgementDocumentType" >  
  <xsd:annotation>  
    ...see annotation...  
  </xsd:annotation>  
  <xsd:sequence>  
    <xsd:element name="MultipleReferencesIndicator" type="udt:IndicatorType"  
      minOccurs="0" >  
      <xsd:annotation>  
        ...see annotation...  
      </xsd:annotation>  
    </xsd:element>  
    <xsd:element name="ID" type="udt:IDType" minOccurs="0" >  
      <xsd:annotation>  
        ...see annotation...  
      </xsd:annotation>  
    </xsd:element>  
    <xsd:element name="TypeCode" type="qdt:DocumentTypeCode" minOccurs="0"  
      maxOccurs="unbounded">  
      <xsd:annotation>  
        ...see annotation...  
      </xsd:annotation>  
    </xsd:element>  
    ...  
  </xsd:sequence>  
</xsd:complexType>
```

1477

#### Example 7-10: Choice

1478  
1479  
1480  
1481  
1482

```
<xsd:complexType name="LocationType">  
  <xsd:annotation>  
    ... see annotation ...  
  </xsd:annotation>  
  <xsd:choice>
```

```

1483     <xsd:element name="GeoCoordinate" type="ram:GeoCoordinateType"
1484       minOccurs="0">
1485       <xsd:annotation>
1486         ... see annotation ...
1487       </xsd:annotation>
1488     </xsd:element>
1489     <xsd:element name="Address" type="ram:AddressType"
1490       minOccurs="0">
1491       <xsd:annotation>
1492         ... see annotation ...
1493       </xsd:annotation>
1494     </xsd:element>
1495     <xsd:element name="Location" type="ram:LocationType"
1496       minOccurs="0">
1497       <xsd:annotation>
1498         ... see annotation ...
1499       </xsd:annotation>
1500     </xsd:element>
1501   </xsd:choice>
1502 </xsd:complexType>

```

1503 **Example 7-11: Sequence + Choice within Object Class "PeriodType"**

```

1504 <xsd:complexType name="PeriodType">
1505   ...
1506   <xsd:sequence>
1507     <xsd:element name="DurationDateTime"
1508       type="qdt:DurationDateTimeType" minOccurs="0"
1509       maxOccurs="unbounded">
1510       ...
1511     </xsd:element>
1512     ...
1513   <xsd:choice>
1514     <xsd:sequence>
1515       <xsd:element name="StartTime" type="udt:TimeType"
1516         minOccurs="0">
1517         ...
1518       </xsd:element>
1519       <xsd:element name="EndTime" type="udt:TimeType"
1520         minOccurs="0">
1521         ...
1522       </xsd:element>
1523     </xsd:sequence>
1524     <xsd:sequence>
1525       <xsd:element name="StartDate" type="udt:DateType"
1526         minOccurs="0">
1527         ...
1528       </xsd:element>
1529       <xsd:element name="EndDate" type="udt:DateType"
1530         minOccurs="0">
1531         ...
1532       </xsd:element>
1533     </xsd:sequence>
1534     <xsd:sequence>
1535       <xsd:element name="StartDateTime" type="udt:DateTimeType"
1536         minOccurs="0">
1537         ...
1538       </xsd:element>
1539       <xsd:element name="EndDateTime" type="udt:DateTimeType"
1540         minOccurs="0">
1541         ...
1542       </xsd:element>
1543     </xsd:sequence>
1544   </xsd:choice>
1545 </xsd:sequence>
1546 </xsd:complexType>

```

---

1547 [R101] The order and cardinality of the elements within an ABIE `xsd:complexType` MUST be  
1548 according to the structure of the ABIE as defined in the model.

---

1549

1550 **Example 7-12: Type definition of an ABIE**

```
1551 <!-- ===== -->
1552 <!-- ===== Type Definitions ===== -->
1553 <!-- ===== -->
1554 <xsd:complexType name="AgriculturalPlotType" >
1555   <xsd:annotation>
1556     ... see annotation ...
1557   </xsd:annotation>
1558   <xsd:sequence>
1559     <xsd:element name="ID" type="udt:IDType" >
1560       <xsd:annotation>
1561         ... see annotation ...
1562       </xsd:annotation>
1563     </xsd:element>
1564     ... see element declaration ...
1565   </xsd:sequence>
1566 </xsd:complexType>
```

1567 **7.3.5 Element Declarations and References**

1568 Every ABIE will have a globally declared element.

- 
- 1569 [R102] For each ABIE, a named **xsd:element** MUST be globally declared.
  - 1570 [R103] The name of the ABIE **xsd:element** MUST be the **cts:DictionaryEntryName** with the  
1571 separators and 'Details' suffix removed and approved abbreviations and acronyms applied.
  - 1572 [R104] Every ABIE global element declaration MUST be of the **xsd:complexType** that represents the  
1573 ABIE.
- 

1574 The content model of the complex type definitions will include both element declarations for BBIEs and  
1575 ASBIEs whose **cts:AssociationType** is Composition, and element references to the globally  
1576 declared elements for ASBIEs whose **cts:AssociationType** is not Composition. The BBIEs will  
1577 always be declared locally.

- 
- 1578 [R105] For every BBIE identified in an ABIE, a named **xsd:element** MUST be locally declared within  
1579 the **xsd:complexType** representing that ABIE.
  - 1580 [R106] Each BBIE element name declaration MUST be the property term and qualifiers and the  
1581 representation term of the basic business information entity (BBIE). Where the word 'identification'  
1582 is the final word of the property term and the representation term is 'identifier', the term  
1583 'identification' MUST be removed. Where the word 'indication' is the final word of the property  
1584 term and the representation term is 'indicator', the term 'indication' MUST be removed from the  
1585 property term.
  - 1586 [R107] If the representation term of a BBIE is 'text', 'text' MUST be removed.
  - 1587 [R108] The BBIE element MUST be based on an appropriate data type that is defined in the UN/CEFACT  
1588 **qdt:QualifiedDataType** or **udt:UnqualifiedDataType** schema modules.
- 

1589 The ASBIEs whose **cts:AssociationType** is Composition will always be declared locally.

- 
- 1590 [R109] For every ASBIE whose **cts:AssociationType** is a composition, a named **xsd:element**  
1591 MUST be locally declared.
  - 1592 [R110] For each locally declared ASBIE, the element name MUST be the ASBIE property term and  
1593 qualifier term(s) and the object class term and qualifier term(s) of the associated ABIE.
  - 1594 [R111] For each locally declared ASBIE, the element declaration MUST be of the **sd:complexType** that  
1595 represents its associated ABIE.
- 

1596 For each ASBIE whose **cts:AssociationType** is not a composition, the globally declared element for the  
1597 associated ABIE will be included in the content model of the associating ASBIE.

- 
- 1598 [R112] For every ASBIE whose **cts:AssociationType** is not a composition, the globally  
1599 declared element for the associated ABIE must be referenced using **xsd:ref**.
-

1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608

### Example 7-13: Element declaration and reference within an ABIE type definition

```
<xsd:complexType name="PurchaseOrderRequestType">  
  <xsd:sequence>  
    <xsd:element name="ID" type="udt:IDType"/>  
    <xsd:element name="SellerParty" type="ram:SellerPartyType"/>  
    <xsd:element name="BuyerParty" type="ram:BuyerPartyType"/>  
    <xsd:element ref="ram:OrderedLineItem" maxOccurs="unbounded"/>  
  </xsd:sequence>  
</xsd:complexType>
```

1609

## 7.3.6 Annotation

1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639

[R113] For every ABIE `xsd:complexType` and `xsd:element` definition a structured set of annotations MAY be present in the following pattern:

- UniqueID (required): The identifier that references an ABIE instance in a unique and unambiguous way.
- Acronym (required): The abbreviation of the type of component. In this case the value will always be ABIE.
- DictionaryEntryName (required): The official name of an ABIE.
- Version (required): An indication of the evolution over time of an ABIE instance.
- Definition (required): The semantic meaning of an ABIE.
- ObjectClassTerm (required): The Object Class Term of the ABIE.
- ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the ABIE.
- BusinessProcessContextValue (optional, repetitive): The business process with which this ABIE is associated.
- GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for this ABIE.
- OfficialConstraintContextValue (optional, repetitive): The official constraint context for this ABIE.
- ProductContextValue (optional, repetitive): The product context for this ABIE.
- IndustryContextValue (optional, repetitive): The industry context for this ABIE.
- BusinessProcessRoleContextValue (optional, repetitive): The role context for this ABIE.
- SupportingRoleContextValue (optional, repetitive): The supporting role context for this ABIE.
- SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for this ABIE.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ABIE.
- BusinessTerm (optional, repetitive): A synonym term under which the ABIE is commonly known and used in the business.
- Example (optional, repetitive): Example of a possible value of an ABIE.

[R114] This rule was combined with [R113].

1640

### Example 7-14: Annotation of an ABIE

1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659

```
<xsd:complexType name="AgriculturalPlotType" >  
  <xsd:annotation>  
    <xsd:documentation xml:lang="en">  
      <ccts:UniqueID>UN01002651</ccts:UniqueID>  
      <ccts:Acronym>ABIE</ccts:Acronym>  
      <ccts:DictionaryEntryName>Agricultural_ Plot. Details</ccts:DictionaryEntryName>  
  
      <ccts:Version>1.0</ccts:Version>  
      <ccts:Definition>A small piece of land used in agriculture.</ccts:Definition>  
      <ccts:ObjectClassTerm>Plot</ccts:ObjectClassTerm>  
      <ccts:ObjectClassQualifierTerm>Agricultural</ccts:ObjectClassQualifierTerm>  
      <ccts:BusinessProcessContextValue>Crop Data Sheet</ccts:BusinessProcessContextValue>  
      <ccts:GeopoliticalorRegionalContextValue>Global</ccts:  
      GeopoliticalorRegionalContextValue >  
      <ccts:OfficialConstraintContextValue>European, National, Local  
      Regulations</ccts:OfficialConstraintContextValue>  
      <ccts:ProductContextValue>Arable crop</ccts:ProductContextValue>  
      <ccts:IndustryContextValue>Agricultural</ccts:IndustryContextValue>  
      <ccts:BusinessProcessRoleContextValue>In All  
      Contexts</ccts:BusinessProcessRoleContextValue>
```

```

1660         <ccts:SupportingRoleContextValue>In All Contexts</ccts:SupportingRoleContextValue>
1661         <ccts:SystemCapabilitiesContextValue>In All
1662         Contexts</ccts:SystemCapabilitiesContextValue>
1663     </xsd:documentation>
1664 </xsd:annotation>
1665     ...
1666 </xsd:complexType>

```

[R115] For every BBIE **xsd:element** declaration a structured set of annotations MAY be present in the following pattern:

- UniqueID (required): The identifier that references a BBIE instance in a unique and unambiguous way.
- Acronym (required): The abbreviation of the type of component. In this case the value will always be BBIE.
- DictionaryEntryName (required): The official name of the BBIE.
- VersionID (required): An indication of the evolution over time of a BBIE instance.
- Definition (required): The semantic meaning of the BBIE.
- Cardinality (required): Indication whether the BBIE Property represents a not-applicable, optional, required and/or repetitive characteristic of the ABIE.
- ObjectClassTerm (required): The Object Class Term of the parent ABIE.
- ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the parent ABIE.
- PropertyTerm (required): The Property Term of the BBIE.
- PropertyQualifierTerm (optional): Qualifies the Property Term of the BBIE.
- PrimaryRepresentationTerm (required): The Primary Representation Term of the BBIE.
- BusinessProcessContextValue (optional, repetitive): The business process with which this BBIE is associated.
- GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for this BBIE.
- OfficialConstraintContextValue (optional, repetitive): The official constraint context for this BBIE.
- ProductContextValue (optional, repetitive): The product context for this BBIE.
- IndustryContextValue (optional, repetitive): The industry context for this BBIE.
- BusinessProcessRoleContextValue (optional, repetitive): The role context for this BBIE.
- SupportingRoleContextValue (optional, repetitive): The supporting role context for this BBIE.
- SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for this BBIE.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to this BBIE.
- BusinessTerm (optional, repetitive): A synonym term under which the BBIE is commonly known and used in the business.
- Example (optional, repetitive): Example of a possible value of a BBIE.

#### Example 7-15: Annotation of a BBIE

```

1700
1701 <xsd:element name="ID" type="udt:IDType" >
1702   <xsd:annotation>
1703     <xsd:documentation xml:lang="en">
1704       <ccts:UniqueID>UN01002652</ccts:UniqueID>
1705       <ccts:Acronym>BBIE</ccts:Acronym>
1706       <ccts:DictionaryEntryName>Agricultural_ Plot. Identification.
1707       Identifier</ccts:DictionaryEntryName>
1708       <ccts:Version>1.0</ccts:Version>
1709       <ccts:Definition>The unique identifier for this agricultural plot.
1710       </ccts:Definition>
1711       </ccts:Cardinality>1</ccts:Cardinality>
1712       <ccts:ObjectClassTerm>Plot</ccts:ObjectClassTerm>

```



```

1713 <ccts:ObjectClassQualifierTerm>Agricultural</ccts:ObjectClassQualifierTerm>
1714 <ccts:PropertyTerm>Identification</ccts:PropertyTerm>
1715 <ccts:PrimaryRepresentationTerm>Identifier</ccts:PrimaryRepresentationTerm>
1716 <ccts:BusinessProcessContextValue>Crop Data
1717 Sheet</ccts:BusinessProcessContextValue>
1718 <ccts:GeopoliticalOrRegionContextValue>Global</ccts:GeopoliticalOrRegionCon
1719 textValue>
1720 <ccts:OfficialConstraintContextValue>European, National, Local
1721 Regulations</ccts:OfficialConstraintContextValue>
1722 <ccts:ProductContextValue>Arable crop</ccts:ProductContextValue>
1723 <ccts:IndustryContextValue>Agricultural</ccts:IndustryContextValue>
1724 <ccts:BusinessProcessRoleContextValue>In All
1725 Contexts</ccts:BusinessProcessRoleContextValue>
1726 <ccts:SupportingRoleContextValue>In All
1727 Contexts</ccts:SupportingRoleContextValue>
1728 <ccts:SystemCapabilitiesContextValue>In All
1729 Contexts</ccts:SystemCapabilitiesContextValue>
1730 </xsd:documentation>
1731 </xsd:annotation>
1732 </xsd:element>

```

[R116] For every ASBIE `xsd:element` declaration a structured set of annotations MAY be present in the following pattern:

- UniqueID (required): The identifier that references an ASBIE instance in a unique and unambiguous way.
- Acronym (required): The abbreviation of the type of component. In this case the value will always be ASBIE.
- DictionaryEntryName (required): The official name of the ASBIE.
- Version (required): An indication of the evolution over time of the ASBIE instance.
- Definition (required): The semantic meaning of the ASBIE.
- Cardinality (required): Indication whether the ASBIE Property represents a not-applicable, optional, required and/or repetitive characteristic of the ABIE.
- ObjectClassTerm (required): The Object Class Term of the associating ABIE.
- ObjectClassQualifierTerm (optional): A term that qualifies the Object Class Term of the associating ABIE.
- PropertyTerm (required): The Property Term of the ASBIE.
- PropertyQualifierTerm (Optional): A term that qualifies the Property Term of the ASBIE.
- AssociatedObjectClassTerm (required): The Object Class Term of the associated ABIE.
- AssociatedObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the associated ABIE.
- AssociationType (required): The Association Type of the ASBIE.
- BusinessProcessContextValue (optional, repetitive): The business process with which this ASBIE is associated.
- GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for this ASBIE.
- OfficialConstraintContextValue (optional, repetitive): The official constraint context for this ASBIE.
- ProductContextValue (optional, repetitive): The product context for this ASBIE.
- IndustryContextValue (optional, repetitive): The industry context for this ASBIE.
- BusinessProcessRoleContextValue (optional, repetitive): The role context for this ASBIE.
- SupportingRoleContextValue (optional, repetitive): The supporting role context for this ASBIE.
- SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for this ASBIE.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ASBIE.

- 1768 ○ BusinessTerm (optional, repetitive): A synonym term under which the ASBIE is commonly
- 1769 known and used in the business.
- 1770 ○ Example (optional, repetitive): Example of a possible value of an ASBIE.

1771 **Example 7-16: Annotation of an ASBIE**

```

1772 <xsd:element name="IncludedInAgriculturalCountrySubDivision" type="ram:
1773 AgriculturalCountrySubDivisionType" minOccurs="0" maxOccurs="unbounded">
1774 <xsd:annotation>
1775 <xsd:documentation xml:lang="en">
1776 <ccts:UniqueID>UN01002659</ccts:UniqueID>
1777 <ccts:Acronym>ASBIE</ccts:Acronym>
1778 <ccts:DictionaryEntryName>Agricultural_ Plot. Included In. Agricultural_
1779 Country Sub-Division</ccts:DictionaryEntryName>
1780 <ccts:Version>1.0</ccts:Version>
1781 <ccts:Definition>An agricultural country sub-division in which this
1782 agricultural plot is included.</ccts:Definition>
1783 <ccts:Cardinality>0..n</ccts:Cardinality>
1784 <ccts:ObjectClassTerm>Plot</ccts:ObjectClassTerm>
1785 <ccts:ObjectClassQualifierTerm>Agricultural</ccts:ObjectClassQualifierTe
1786 rm>
1787 <ccts:AssociationType>composition</ccts:AssociationType>
1788 <ccts:PropertyTerm>Included In</ccts:PropertyTerm>
1789 <ccts:AssociatedObjectClassTerm>Country Sub-
1790 Division</ccts:AssociatedObjectClassTerm>
1791 <ccts:AssociatedObjectClassQualifierTerm>Agricultural</ccts:AssociatedOb
1792 jectClassQualifierTerm>
1793 <ccts:BusinessProcessContextValue>Crop Data
1794 Sheet</ccts:BusinessProcessContextValue>
1795 <ccts:GeopoliticalOrRegionContextValue>Global</ccts:GeopoliticalOrRegion
1796 ContextValue>
1797 <ccts:OfficialConstraintContextValue>European, National, Local
1798 Regulations</ccts:OfficialConstraintContextValue>
1799 <ccts:ProductContextValue>Arable crop</ccts:ProductContextValue>
1800 <ccts:IndustryContextValue>Agricultural</ccts:IndustryContextValue>
1801 <ccts:BusinessProcessRoleContextValue>In All
1802 Contexts</ccts:BusinessProcessRoleContextValue>
1803 <ccts:SupportingRoleContextValue>In All
1804 Contexts</ccts:SupportingRoleContextValue>
1805 <ccts:SystemCapabilitiesContextValue>In All
1806 Contexts</ccts:SystemCapabilitiesContextValue></xsd:documentation>
1807 </xsd:annotation>
1808 </xsd:element>

```

1809 **7.4 Core Component Type**

1810 **7.4.1 Use of Core Component Type Module**

1811 The purpose of the core component type module is to define the core component types on which the  
 1812 unqualified data types are based. This module is only for reference and will not be included/imported in any  
 1813 schema.

1814 **7.4.2 Schema Construct**

1815 The core component type schema module will be constructed in a standardized format in order to ensure  
 1816 consistency and ease of use. The specific format is shown below and must adhere to the format of the  
 1817 relevant sections as detailed in Appendix B.

1818 **Example 7-17: Structure of Core Component Type Schema Module**

```

1819 <?xml version="1.0" encoding="utf-8"?>
1820 <!-- ===== -->
1821 <!-- ===== Core Component Type Schema Module ===== -->
1822 <!-- ===== -->
1823 <!--
1824 Scheme agency: UN/CEFACT
1825 Scheme version: 2.0
1826 Schema date: [SCHEMADATE]
1827 ... see intellectual property disclaimer ...

```

1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838

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

1839

### 7.4.3 Namespace Scheme

1840

[R117] The core component type (CCT) schema module MUST be represented by the token `cct`.

1841

#### Example 7-18: Namespace of Core Component Type Schema Module

1842

```
"urn:un:unece:uncefact:documentation:draft:CoreComponentType:2"
```

1843

#### Example 7-19: Schema-element of Core Component Type Schema Module

1844  
1845  
1846  
1847  
1848

```
<xsd:schema
  targetNamespace="urn:un:unece:uncefact:documentation:draft:CoreComponentType:2"
  xmlns:cct="urn:un:unece:uncefact:documentation:draft:CoreComponentType:2"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
```

1849

### 7.4.4 Imports and Includes

1850

The core component type schema module does not import or include any other schema modules.

1851

[R118] The `cct:CoreCoreComponentType` schema module MUST NOT include or import any other schema modules.

1852

1853

### 7.4.5 Type Definitions

1854

[R119] Every core component type MUST be defined as a named `xsd:complexType` in the `cct:CoreComponentType` schema module.

1855

1856

[R120] The name of each `xsd:complexType` based on a core component type MUST be the dictionary entry name of the core component type (CCT), with the separators and spaces removed and approved abbreviations applied.

1857

1858

1859

[R121] Each core component type `xsd:complexType` definition MUST contain one `xsd:simpleContent` element.

1860

1861

[R122] The core component type `xsd:complexType` definition `xsd:simpleContent` element MUST contain one `xsd:extension` element. This `xsd:extension` element must include an XSD based attribute that defines the specific XSD built-in data type required for the CCT content component.

1862

1863

1864

1865

[R123] Within the core component type `xsd:extension` element a `xsd:attribute` MUST be declared for each supplementary component pertaining to that core component type.

1866

1867

#### Example 7-20: Type definition of a CCT

1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880

```
<!-- ===== Type Definitions ===== -->
<!-- ===== AmountType ===== -->
<xsd:complexType name="AmountType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
      <xsd:attribute name="currencyID" type="xsd:token" use="optional" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

1881  
1882  
1883  
1884  
1885  
1886  
1887

```
        ... see annotation ...  
    </xsd:annotation>  
</xsd:attribute>  
    ... see attribute declaration ...  
</xsd:extension>  
</xsd:simpleContent>  
</xsd:complexType>
```

1888

## 7.4.6 Attribute Declarations

1889  
1890  
1891  
1892  
1893  
1894

The current CCTS does not specify the components of the CCT supplementary component dictionary entry name. However, in order to ensure a standard approach to declaring the supplementary components as attributes, BPS has applied the naming concepts from ISO 11179, part 5. Specifically, BPS has defined the dictionary entry name as it is stated in CCTS in terms of object class, property term, and representation term. These components are identified in the annotation documentation for each supplementary component in the CCT schema module.

1895  
1896  
1897

[R124] Each core component type supplementary component **xsd:attribute** name MUST be the CCTS supplementary component dictionary entry name with the separators and spaces removed.

1898  
1899  
1900

[R125] If the object class of the supplementary component dictionary entry name contains the name of the representation term of the parent CCT, the duplicated object class word or words MUST be removed from the supplementary component **xsd:attribute** name.

1901  
1902  
1903

[R126] If the object class of the supplementary component dictionary entry name contains the term 'identification', the term 'identification' MUST be removed from the supplementary component **xsd:attribute** name.

1904  
1905  
1906

[R127] If the representation term of the supplementary component dictionary entry name is 'text', the representation term MUST be removed from the supplementary component **xsd:attribute** name.

1907  
1908

[R128] The attribute representing the supplementary component MUST be based on the appropriate XSD built-in data type.

1909

### Example 7-21: Supplementary component other than code or identifier

1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920

```
<xsd:complexType name="BinaryObjectType">  
    ...  
    <xsd:simpleContent>  
        <xsd:extension base="xsd:base64Binary">  
            <xsd:attribute name="format" type="xsd:string" use="optional">  
                ...  
            </xsd:attribute>  
            ...  
        </xsd:extension>  
    </xsd:simpleContent>  
</xsd:complexType>
```

1921

## 7.4.7 Extension and Restriction

1922  
1923

The core component type schema module is a generic module based on the underlying core component types. No restriction or extension is appropriate.

1924

## 7.4.8 Annotation

1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933

[R129] For every core component type **xsd:complexType** definition a structured set of annotations MAY be present in the following pattern:

- UniqueID (required): The identifier that references the Core Component Type instance in a unique and unambiguous way.
- Acronym (required): The abbreviation of the type of component. . In this case the value will always be CCT.
- DictionaryEntryName (required): The official name of a Core Component Type.
- Version (required): An indication of the evolution over time of a Core Component Type instance.

- 1934 ○ Definition (required): The semantic meaning of a Core Component Type.
- 1935 ○ PrimaryRepresentationTerm (required): The primary representation term of the Core
- 1936 Component Type.
- 1937 ○ PrimitiveType (required): The primitive data type of the Core Component Type.
- 1938 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 1939 applicable to the Core Component Type.

#### 1940 Example 7-22: Annotation of a CCT

```

1941 ... see type definition ...
1942 <xsd:annotation>
1943   <xsd:documentation xml:lang="en">
1944     <ccts:UniqueID>UNDT000001</ccts:UniqueID>
1945     <ccts:Acronym>CCT</ccts:Acronym>
1946     <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>
1947     <ccts:Version>1.0</ccts:Version>
1948     <ccts:Definition>A number of monetary units specified in a currency
1949       where the unit of the currency is explicit or
1950       implied.</ccts:Definition>
1951     <ccts:PrimaryRepresentationTerm>Amount</ccts:PrimaryRepresentationTerm>
1952     <ccts:PrimitiveType>decimal</ccts:PrimitiveType>
1953   </xsd:documentation>
1954 </xsd:annotation>
1955 ... see type definition ...

```

1956 [R130] For every supplementary component **xsd:attribute** declaration a structured set of

1957 annotations MAY be present in the following pattern:

- 1958 ○ UniqueID (optional): The identifier that references the Supplementary Component instance in
- 1959 a unique and unambiguous way.
- 1960 ○ Acronym (required): The abbreviation of the type of Supplementary Component. In this case
- 1961 the value will always be SC.
- 1962 ○ DictionaryEntryName (required): The official name of the Supplementary Component.
- 1963 ○ Definition (required): The semantic meaning of the Supplementary Component.
- 1964 ○ Cardinality (required): The cardinality of the Supplementary Component.
- 1965 ○ ObjectClassTerm (required): The Object Class of the Supplementary Component.
- 1966 ○ PropertyTerm (required): The Property Term of the Supplementary Component.
- 1967 ○ PrimaryRepresentationTerm (required): The Primary Representation Term of the
- 1968 Supplementary Component.
- 1969 ○ PrimitiveType (required): The primitive data type of the Supplementary Component.
- 1970 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 1971 applicable to the Supplementary Core Component.

#### 1972 Example 7-23: Annotation of a supplementary component

```

1973 ... see attribute declaration ...
1974 <xsd:annotation>
1975   <xsd:documentation xml:lang="en">
1976     <ccts:Acronym>SC</ccts:Acronym>
1977     <ccts:DictionaryEntryName>Amount. Currency. Identifier</ccts:DictionaryEntryName>
1978     <ccts:Definition>The currency of the amount.</ccts:Definition>
1979     <ccts:ObjectClassTerm>Amount</ccts:ObjectClassTerm>
1980     <ccts:PropertyTerm>Currency</ccts:PropertyTerm>
1981     <ccts:PrimaryRepresentationTerm>Identifier</ccts:PrimaryRepresentationTerm>
1982     <ccts:PrimitiveType>string</ccts:PrimitiveType>
1983   </xsd:documentation>
1984 </xsd:annotation>
1985 ... see attribute declaration ...

```

## 1986 7.5 Unqualified Data Type

### 1987 7.5.1 Use of Unqualified Data Type Module

1988 The unqualified data type schema module will define data types for all primary and secondary representation

1989 terms as specified in the CCTS. All data types will be defined as **xsd:complexType** or **xsd:simpleType**

1990 and will only reflect restrictions as specified in CCTS and agreed upon industry best practices.

1991 **7.5.2 Schema Construct**

1992 The unqualified data types schema will be constructed in a standardized format in order to ensure  
1993 consistency and ease of use. The specific format is shown below and must adhere to the format of the  
1994 relevant sections as detailed in Appendix B.

1995 **Example 7-24: Structure of unqualified data type schema module**

```
1996 <?xml version="1.0" encoding="utf-8"?>  
1997 <!-- =====>  
1998 <!-- ===== Unqualified Data Type Schema Module =====>  
1999 <!-- =====>  
2000 <!--  
2001 Schema agency: UN/CEFACT  
2002 Schema version: 2.0  
2003 Schema date: [SCHEMADATE]  
  
2004 ... see intellectual property disclaimer ...  
  
2005 -->  
2006 <xsd:schema targetNamespace=  
2007 ... see namespace ...  
2008 xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"  
2009 attributeFormDefault="unqualified">  
2010 <!-- =====>  
2011 <!-- ===== Imports =====>  
2012 <!-- =====>  
2013 ... see imports ...  
2014 <!-- =====>  
2015 <!-- ===== Type Definitions =====>  
2016 <!-- =====>  
2017 <!-- ===== Amount. Type =====>  
2018 <!-- =====>  
2019 <xsd:complexType name="AmountType">  
2020 ... see type definition ...  
2021 </xsd:complexType>  
2022 ...  
2023 </xsd:schema>
```

2024 **7.5.3 Namespace Scheme**

2025 [R131] The Unqualified Data Type schema module namespace MUST be represented by the token `udt`.

2026 **Example 7-25: Namespace of unqualified data type schema module**

```
2027 "urn:un:unece:unefact:data:draft:UnqualifiedDataType:1"
```

2028 **Example 7-26: Schema-element of unqualified data type schema module**

```
2029 <xsd:schema targetNamespace=  
2030 "urn:un:unece:unefact:data:draft:UnqualifiedDataType:1"  
2031 xmlns:udt=  
2032 "urn:un:unece:unefact:data:draft:UnqualifiedDataType:1"  
2033 xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2034 elementFormDefault="qualified" attributeFormDefault="unqualified">
```

2035 **7.5.4 Imports and Includes**

2036 To maximize reusability and minimize maintenance costs, it is strongly discouraged that the Unqualified  
2037 Data Type schema will import any code list or identifier list schema modules. The schema may indicate that  
2038 certain code or identifier lists are recommended for use by certain supplementary components, but those  
2039 code lists or schema modules should be bound only if absolutely necessary. The Unqualified Data Type  
2040 schema will not import any other schema modules.

2041 [R132] The `udt:UnqualifiedDataType` schema MUST only import the following schema  
2042 modules:  
2043 – `ids:IdentifierList` schema modules  
2044 – `clm:CodeList` schema modules



2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054

### Example 7-27: Imports

```
<!-- ===== Imports ===== -->
<!-- ===== -->
<!-- ===== Imports of Code Lists ===== -->
<!-- ===== -->
<xsd:import namespace=
  "urn:un:unece:unefact:codelist:standard:6:3055:D12A"
  schemaLocation="www.unece.org/unefact/codelist/standard/UNECE_AgencyIdentificationCode_
D12A.xsd"/>
```

2055  
2056  
2057  
2058

## 7.5.5 Type Definitions

Each unqualified data type is represented in the unqualified data type schema module as either a **xsd:complexType** or a **xsd:simpleType**. Unqualified data types are defined based on the core component types as specified in the CCTS.

2059  
2060

---

[R133] An unqualified data type MUST be defined for each approved primary and secondary representation terms identified in the CCTS Permissible Representation Terms table.

2061  
2062  
2063

[R134] The name of each unqualified data type MUST be the dictionary entry name of the primary or secondary representation term, with the word 'Type' appended, the separators and spaces removed and approved abbreviations applied.

---

2064  
2065  
2066

In accordance with rules and principles in this document, the unqualified data type will be based on XSD built-in data types *whenever the XSD built-in data type meets the functionality of all the supplementary components for that data type*.

---

2067  
2068  
2069

[R135] For every unqualified data type whose supplementary components map directly to the properties of a XSD built-in data type, the unqualified data type MUST be defined as a named **xsd:simpleType** in the **udt:UnqualifiedDataType** schema module.

2070  
2071  
2072

[R136] Every unqualified data type **xsd:simpleType** MUST contain one **xsd:restriction** element. This **xsd:restriction** element MUST include an **xsd:base** attribute that defines the specific XSD built-in data type required for the content component.

---

2073  
2074  
2075  
2076

When the unqualified data type does not directly map to an **xsd:simpleType** due to the supplementary components needing to be expressed, the unqualified data type will be defined as an **xsd:complexType**. If, however, some implementers want to use the simple type but others want to use the complex type, the unqualified data type should allow a choice between the two, using **xsd:choice**.

---

2077  
2078  
2079

[R137] For every unqualified data type whose supplementary components are not equivalent to the properties of a XSD built-in data type, the unqualified data type MUST be defined as an **xsd:complexType** in the **udt:UnqualifiedDataType** schema module.

2080  
2081

[R138] Every unqualified data type **xsd:complexType** definition MUST contain one **xsd:simpleContent** element.

2082  
2083  
2084

[R139] Every unqualified data type **xsd:complexType** **xsd:simpleContent** element MUST contain one **xsd:extension** element. This **xsd:extension** element must include an **xsd:base** attribute that defines the specific XSD built-in data type required for the content component.

2085  
2086  
2087

[R204] When a combination of the complex and simple types are necessary to support business requirements, the element MUST be declared as an **xsd:complexType** with an **xsd:choice** between elements declared as the two different alternatives.

---

2088

## 7.5.6 Attribute Declarations

2089  
2090  
2091  
2092  
2093

Each core component supplementary component is declared as an attribute of the complex type. In certain circumstances, continually providing the attributes necessary to convey code and identifier list metadata for multiple repetitions of the same element may prove burdensome. The namespace scheme for code lists and identification scheme lists has been specifically designed to include some of the supplementary components for the CCTs **Code. Type** and **Identifier. Type**. If an implementation desires this metadata to be



2094 conveyed as attributes rather than part of the namespace declaration, a qualified data type with the additional  
2095 attributes representing the missing supplementary components can be specified.

---

2096 [R140] Within the unqualified data type `xsd:complexType` `xsd:extension` element an  
2097 `xsd:attribute` MUST be declared for each supplementary component pertaining to the  
2098 underlying CCT.

---

2099 The attributes representing supplementary components will be named based on their underlying CCT  
2100 supplementary component. The user declared attributes can be based on:

- 2101 • XSD built-in types, if a specific supplementary component represents a variable value,
- 2102 • Simple types of a code list, if the specific supplementary component represents a code value, or
- 2103 • Simple types of an identifier scheme, if the specific supplementary component represents an identifier  
2104 value.

2105 For some CCTs, the CCTS identifies restrictions in the form of pointing to certain restrictive code or identifier  
2106 lists. These restrictive lists will be declared in the code list or identifier schema module and the unqualified  
2107 data type may reference these lists.

---

2108 [R141] Each supplementary component `xsd:attribute` name MUST be the supplementary  
2109 component name with the separators and spaces removed, and approved abbreviations and  
2110 acronyms applied.

2111 [R142] If the object class of the supplementary component dictionary entry name contains the name of  
2112 the representation term, the duplicated object class word or words MUST be removed from the  
2113 supplementary component `xsd:attribute` name.

2114 [R143] If the object class of the supplementary component dictionary entry name contains the term  
2115 'identification', the term 'identification' MUST be removed from the supplementary component  
2116 `xsd:attribute` name.

2117 [R144] If the representation term of the supplementary component dictionary entry name is 'text', the  
2118 representation term MUST be removed from the supplementary component `xsd:attribute`  
2119 name.

---

## 2120 Example 7-28: Type definitions of unqualified data types

```
2121 <!-- ===== Type Definitions ===== -->
2122 <!-- =====
2123 <!-- ===== Amount. Type ===== -->
2124 <!-- =====
2125 <xsd:complexType name="AmountType">
2126   <xsd:annotation>
2127     ... see annotation ...
2128   </xsd:annotation>
2129   <xsd:simpleContent>
2130     <xsd:extension base="xsd:decimal">
2131       <xsd:attribute name="currencyID"
2132         type="clm5ISO42173A:ISO3AlphaCurrencyCodeContentType" use="optional">
2133         <xsd:annotation>
2134           ... see annotation ...
2135         </xsd:annotation>
2136       </xsd:attribute>
2137       <xsd:attribute name="currencyCodeListVersionID" type="xsd:token" use="optional">
2138         <xsd:annotation>
2139           ... see annotation ...
2140         </xsd:annotation>
2141       </xsd:attribute>
2142     </xsd:extension>
2143   </xsd:simpleContent>
2144 </xsd:complexType>
2145 <!-- =====
2146 <!-- ===== Binary Object. Type ===== -->
2147 <!-- =====
2148 <xsd:complexType name="BinaryObjectType">
2149   <xsd:annotation>
2150     ... see annotation ...
2151   </xsd:annotation>
2152   <xsd:simpleContent>
```

```

2153     <xsd:extension base="xsd:base64Binary">
2154       <xsd:attribute name="mimeCode" type="clmIANA MIMEMediaType:MIMEMediaTypeContentType">
2155         <xsd:annotation>
2156           ... see annotation ...
2157         </xsd:annotation>
2158       </xsd:attribute>
2159       <xsd:attribute name="encodingCode"
2160         type="clm60133:CharacterSetEncodingCodeContentType" use="optional">
2161         <xsd:annotation>
2162           ... see annotation ...
2163         </xsd:annotation>
2164       </xsd:attribute>
2165       <xsd:attribute name="characterSetCode"
2166         type="clmIANACharacterSetCode:CharacterSetCodeContentType"
2167         use="optional">
2168         <xsd:annotation>
2169           ... see annotation ...
2170         </xsd:annotation>
2171       </xsd:attribute>
2172       <xsd:attribute name="uri" type="xsd:anyURI" use="optional">
2173         <xsd:annotation>
2174           ... see annotation ...
2175         </xsd:annotation>
2176       </xsd:attribute>
2177       <xsd:attribute name="filename" type="xsd:string" use="optional">
2178         <xsd:annotation>
2179           ... see annotation ...
2180         </xsd:annotation>
2181       </xsd:attribute>
2182     </xsd:extension>
2183   </xsd:simpleContent>
2184 </xsd:complexType>

```

2185 The user declared attributes are dependent on the type of representation term of the specific supplementary  
2186 component.

---

[R145] If the representation term of the supplementary component is 'Code' and validation is required, then the attribute representing this supplementary component MUST be based on the defined **xsd:simpleType** of the appropriate external imported code list.

---

#### 2190 Example 7-29: Supplementary Component is a Code

```

2191 <xsd:complexType name="MeasureType">
2192   <xsd:simpleContent>
2193     <xsd:extension base="xsd:decimal">
2194       <xsd:attribute name="unitCode"
2195         type="clm6Recommendation20:MeasurementUnitCommonCodeContentType" use="optional">
2196         ...
2197       </xsd:attribute>
2198       ...
2199     </xsd:extension>
2200   </xsd:simpleContent>
2201 </xsd:complexType>

```

---

[R146] If the representation term of the supplementary component is 'Identifier' and validation is required, then the attribute representing this supplementary component MUST be based on the defined **xsd:simpleType** of the appropriate external imported identifier list.

---

#### 2205 Example 7-30: Supplementary component is an identifier

```

2206 <xsd:complexType name="AmountType">
2207   <xsd:annotation>
2208     ...
2209   </xsd:annotation>
2210   <xsd:simpleContent>
2211     <xsd:extension base="xsd:decimal">
2212       <xsd:attribute name="currencyID"
2213         type="clm5ISO42173A:ISO3AlphaCurrencyCodeContentType" use="optional">
2214         ...
2215       </xsd:attribute>
2216     </xsd:extension>
2217   </xsd:simpleContent>
2218 </xsd:complexType>

```

---

2219 [R147] If the representation term of the supplementary component is other than 'Code' or 'Identifier',  
2220 then the attribute representing this supplementary component MUST be based on the  
2221 appropriate XSD built-in data type.

---

### 2222 Example 7-31: Supplementary component other than code or identifier

```
2223 <xsd:complexType name="BinaryObjectType">  
2224 ...  
2225 <xsd:simpleContent>  
2226 <xsd:extension base="xsd:base64Binary">  
2227 <xsd:attribute name="format" type="xsd:string" use="optional">  
2228 ...  
2229 </xsd:attribute>  
2230 ...  
2231 </xsd:extension>  
2232 </xsd:simpleContent>  
2233 </xsd:complexType>
```

## 2234 7.5.7 Extension and Restriction

2235 The unqualified data types can be further restricted through the creation of qualified data types. These  
2236 qualified data types are defined in the `qdt:QualifiedDataType` schema module.

## 2237 7.5.8 Annotation

---

2238 [R148] For every unqualified data type `xsd:complexType` or `xsd:simpleType` definition a structured  
2239 set of annotations MAY be present in the following pattern:

- 2240 ○ UniqueID (required): The identifier that references an Unqualified Data Type instance in a  
2241 unique and unambiguous way.
- 2242 ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
2243 always be UDT.
- 2244 ○ DictionaryEntryName (required): The official name of the Unqualified Data Type.
- 2245 ○ Version (required): An indication of the evolution over time of the Unqualified Data Type  
2246 instance.
- 2247 ○ Definition (required): The semantic meaning of the Unqualified Data Type.
- 2248 ○ PrimaryRepresentationTerm (required): The primary representation term of the Unqualified  
2249 Data Type.
- 2250 ○ PrimitiveType (required): The primitive data type of the Unqualified Data Type.
- 2251 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are  
2252 applicable to the Unqualified Data Type.

---

### 2253 Example 7-32: Annotation of unqualified type definition

```
2254 .. see complex type definition ...  
2255 <xsd:annotation>  
2256 <xsd:documentation xml:lang="en">  
2257 <ccts:UniqueID>UNDT000001</ccts:UniqueID>  
2258 <ccts:Acronym>UDT</ccts:Acronym>  
2259 <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>  
2260 <ccts:Version>2.01</ccts:Version>  
2261 <ccts:Definition>A number of monetary units specified in a currency where the  
2262 unit of the currency is explicit or implied.</ccts:Definition>  
2263 <ccts:PrimitiveType>decimal</ccts:PrimitiveType>  
2264 </xsd:documentation>  
2265 </xsd:annotation>  
2266 ... see complex type definition ...
```

---

2267 [R149] For every supplementary component `xsd:attribute` declaration a structured set of  
2268 annotations MAY be present in the following pattern:

- 2269 ○ UniqueID (optional): The identifier that references a Supplementary Component instance in a  
2270 unique and unambiguous way.
- 2271 ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
2272 always be SC.
- 2273 ○ Dictionary Entry Name (required): The official name of the Supplementary Component.
- 2274 ○ Definition (required): The semantic meaning of the Supplementary Component.

---

- 2275 ○ Cardinality (mandatory): The cardinality of the Supplementary Component.
- 2276 ○ ObjectClassTerm (mandatory): The Object Class of the Supplementary Component.
- 2277 ○ PropertyTerm (mandatory): The Property Term of the Supplementary Component.
- 2278 ○ PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the
- 2279 Supplementary Component.
- 2280 ○ PrimitiveType (mandatory): The primitive data type of the Unqualified Data Type.
- 2281 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 2282 applicable to the Supplementary Component.

2283 **Example 7-33: Annotation of a supplementary component**

```

2284 ... see complex type definition ...
2285 <xsd:attribute name="currencyID" type="iso4217:CurrencyCodeContentType"
2286 use="required">
2287 <xsd:annotation>
2288 <xsd:documentation xml:lang="en">
2289 <ccts:Acronym>SC</ccts:Acronym>
2290 <ccts:DictionaryEntryName>Amount. Currency. Identifier
2291 </ccts:DictionaryEntryName>
2292 <ccts:Definition>The currency of the amount.</ccts:Definition>
2293 <ccts:Cardinality>0..1</ccts:Cardinality>
2294 <ccts:ObjectClassTerm>Amount</ccts:ObjectClassTerm>
2295 <ccts:PropertyTerm>Currency</ccts:PropertyTerm>
2296 <ccts:PrimaryRepresentationTerm>Identifier<ccts:PrimaryRepresentationTerm>
2297 <ccts:PrimitiveType>decimal</ccts:PrimitiveType>
2298 <ccts:usageRule>By default, use latest version of ISO 4217.</ccts:usageRule>
2299 </xsd:documentation>
2300 </xsd:annotation>
2301 </xsd:attribute>
2302 ... see complex type definition ...

```

2303 **7.6 Qualified Data Type**

2304 Ensuring consistency of qualified data types with the UN/CEFACT modularity and reuse goals requires  
 2305 creating a single schema module that defines all qualified data types. The qualified data type schema module  
 2306 name must follow the UN/CEFACT schema module naming approach. The qualified data type schema  
 2307 module will be used by the reusable ABIE schema module and all root schema modules.

2308 **7.6.1 Use of Qualified Data Type Schema Module**

2309 The data types defined in the unqualified data type schema module are of type **xsd:complexType** or  
 2310 **xsd:simpleType**. These types are intended to be suitable as the **xsd:base** type for some, but not all,  
 2311 BBIEs represented as **xsd:elements**. As business process modelling reveals the need for specialized data  
 2312 types, new qualified types will need to be defined. The qualified data types will also be necessary to define  
 2313 code lists and identifier lists. These new qualified data types must be based on an unqualified data type and  
 2314 must represent a semantic or technical restriction of the unqualified data type. Technical restrictions must be  
 2315 implemented as a **xsd:restriction** or a new **xsd:simpleType** if the supplementary components of the  
 2316 qualified data type map directly to the properties of a XSD built-in data type.

2317 **7.6.2 Schema Construct**

2318 The qualified data type schema will be constructed in a standardized format in order to ensure consistency  
 2319 and ease of use. The specific format is shown below and must adhere to the format of the relevant sections  
 2320 as detailed in Appendix B.

2321 **Example 7-34: Structure of qualified data type schema module**

```

2322 <?xml version="1.0" encoding="utf-8"?>
2323 <!-- ===== -->
2324 <!-- ===== Qualified Data Type Schema Module ===== -->
2325 <!-- ===== -->
2326 <!--
2327 Schema agency: UN/CEFACT
2328 Schema version: 2.0
2329 Schema date: [SCHEMADATE]
2330 ... see intellectual property disclaimer ...

```

```

2331  -->
2332  <xsd:schema targetNamespace=
2333  ... see namespace ... xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2334  elementFormDefault="qualified" attributeFormDefault="unqualified">
2335  <!-- ===== -->
2336  <!-- ===== Imports ===== -->
2337  <!-- ===== -->
2338  ... see imports ...
2339  <!-- ===== -->
2340  <!-- ===== Type Definitions ===== -->
2341  <!-- ===== -->
2342  ... see type definitions ...
2343  </xsd:schema>

```

### 2344 7.6.3 Namespace Scheme

---

2345 [R150] The Qualified Data Type schema module namespace MUST be represented by the token `qdt`.

---

#### 2346 Example 7-35: Namespace name

```
2347 "urn:un:unece:uncefact:data:draft:QualifiedDataType:1"
```

#### 2348 Example 7-36: Schema element

```

2349 <xsd:schema targetNamespace="urn:un:unece:uncefact:data:draft:QualifiedDataType:1"
2350 xmlns:udt="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchema:1"
2351 xmlns:qdt="urn:un:unece:uncefact:data:draft:QualifiedDataTypeSchemaModule:1"
2352 xmlns:xsd="http://www.w3.org/2001/XMLSchema"

```

### 2353 7.6.4 Imports and Includes

2354 Qualified data types will be derived from data types defined in the unqualified data types, code list, and  
 2355 identifier list schema modules. Code or identifier lists should be bound to a qualified data type only when  
 2356 absolutely necessary to avoid introducing complications in the maintenance of implementations.

---

2357 [R151] The `qdt:QualifiedDataType` schema module MUST import the  
 2358 `udt:UnqualifiedDataType` schema module.

2359 [R205] If a coupled design approach is used, then the `qdt:QualifiedDataType` schema module MUST  
 2360 import all code list and identifier scheme schemas used in the module.

---

2361 [Note]

2362 If needed, relevant UN/CEFACT and external code list and identifier scheme schema  
 2363 modules not imported by the `udt:UnqualifiedDataType` schema module may be  
 2364 imported.

### 2365 7.6.5 Type Definitions

---

2366 [R152] Where required to change facets of an existing unqualified data type, a new data type MUST be  
 2367 defined in the `qdt:QualifiedDataType` schema module.

2368 [R153] A qualified data type MUST be based on an unqualified or qualified data type and add some  
 2369 semantic and/or technical restriction to the base data type.

2370 [R154] The name of a qualified data type MUST be the name of its base unqualified or qualified data  
 2371 type with separators and spaces removed and with its qualifier term added.

---

2372 The qualified data types can be derived from an unqualified or qualified data type `xsd:complexType` or  
 2373 `xsd:simpleType` or the code or identifier list schema module content type.

---

2374 [R155] When a qualified data type is based on an unqualified data type that contains an  
 2375 `xsd:choice` element, then the qualified data type MUST be based on one or the other of  
 2376 the elements, but not both.

---

2378

**Example 7-37: Type Definitions**

```

2379 <!-- ===== -->
2380 <!-- ===== Type Definitions ===== -->
2381 <!-- ===== -->
2382 <!-- ===== Qualified Data Type based on DateTime Type ===== -->
2383 <!-- ===== -->
2384 <!-- ===== Formatted_DateTime_Type ===== -->
2385 <!-- ===== -->
2386 <xsd:complexType name="FormattedDateTimeType">
2387   <xsd:annotation>
2388     ... see annotation ...
2389   </xsd:annotation>
2390   <xsd:sequence>
2391     <xsd:simpleContent>
2392       <xsd:extension base="xsd:string">
2393         <xsd:attribute name="format" type="xsd:string">
2394           <xsd:annotation>
2395             ... see annotation ...
2396           </xsd:annotation>
2397         </xsd:attribute>
2398       </xsd:extension>
2399     </xsd:simpleContent>
2400   </xsd:sequence>
2401 </xsd:simpleType>
2402 ...
2403 <!-- ===== -->
2404 <!-- ===== Qualified Data Type based on Identifier_Type ===== -->
2405 <!-- ===== -->
2406 <!-- ===== Country_Identifier_Type ===== -->
2407 <!-- ===== -->
2408 <xsd:complexType name="CountryIDType">
2409   <xsd:annotation>
2410     ... see annotation ...
2411   </xsd:annotation>
2412   <xsd:simpleContent>
2413     <xsd:extension base="ids5ISO316612A:ISOTwoletterCountryCodeContentType">
2414       <xsd:attribute name="schemeID" type="xsd:token" use="optional">
2415         <xsd:annotation>
2416           ... see annotation ...
2417         </xsd:annotation>
2418       </xsd:attribute>
2419       ...
2420     </xsd:extension>
2421   </xsd:simpleContent>
2422 </xsd:complexType>
2423 ...

```

2424 [R156] Every qualified data type based on an unqualified or qualified data type **xsd:complexType**  
 2425 whose supplementary components do not map directly to the properties of a XSD built-in data  
 2426 type

- 2427 MUST be defined as a **xsd:complexType**
- 2428 MUST contain one **xsd:simpleContent** element
- 2429 MUST contain one **xsd:restriction** element
- 2430 MUST include the unqualified data type as its **xsd:base** attribute.

2431 [R157] Every qualified data type based on an unqualified or qualified data type **xsd:simpleType**  
 2432 MUST contain one **xsd:restriction** element  
 2433 MUST include the unqualified or qualified data type as its **xsd:base** attribute or if the facet  
 2434 restrictions can be achieved by use of a XSD built-in data type, then that XSD built-in data  
 2435 type may be used as the **xsd:base** attribute.

2436 [R158] Every qualified data type based on a single code list or identifier list **xsd:simpleType** MUST  
 2437 contain one **xsd:restriction** element or **xsd:union** element. When using the  
 2438 **xsd:restriction** element, the **xsd:base** attribute MUST be set to the code list or identifier list  
 2439 schema module defined simple type with appropriate namespace qualification. When using the  
 2440 **xsd:union** element, the **xsd:member** type attribute MUST be set to the code list or identifier list  
 2441 schema module defined simple types with appropriate namespace qualification.

2442 XML declarations for using code lists in qualified data types are shown in the following examples.

2443 **Example 7-38: Usage of only one Code List**

```
2444 <xsd:simpleType name="TemperatureMeasureUnitCodeType">
2445 <xsd:annotation>
2446 ... see annotation ...
2447 </xsd:annotation>
2448 <xsd:restriction base="clm6Recommendation20:MeasurementUnitCommonCodeContentType">
2449 <xsd:length value="3"/>
2450 <xsd:enumeration value="BTU">
2451 <xsd:annotation>
2452 <xsd:documentation xml:lang="en">
2453 <ccts:Name>British thermal unit</ccts:Name>
2454 </xsd:documentation>
2455 </xsd:annotation>
2456 </xsd:enumeration>
2457 <xsd:enumeration value="CEL">
2458 <xsd:annotation>
2459 <xsd:documentation xml:lang="en">
2460 <ccts:Name>degree Celsius</ccts:Name>
2461 </xsd:documentation>
2462 </xsd:annotation>
2463 </xsd:enumeration>
2464 <xsd:enumeration value="FAH">
2465 <xsd:annotation>
2466 <xsd:documentation xml:lang="en">
2467 <ccts:Name>degree Fahrenheit</ccts:Name>
2468 </xsd:documentation>
2469 </xsd:annotation>
2470 </xsd:enumeration>
2471 </xsd:restriction>
2472 </xsd:simpleType>
```

2473 **Example 7-39: Combination of Code Lists**

```
2474 <xsd:simpleType name="AccountDutyCodeType">
2475 <xsd:annotation>
2476 ... see annotation ...
2477 </xsd:annotation>
2478 <xsd:union memberType="clm64437:AccountTypeCodeContentType
2479 clm65153:DutyTaxFeeTypeCodeContentType"/>
2480 </xsd:simpleType>
```

2481 [R159] Every qualified data type that has a choice of two or more code lists or identifier lists  
2482 MUST be defined as an **xsd:complexType**  
2483 MUST contain the **xsd:choice** element whose content model must consist of element  
2484 references for the alternative code lists or identifier lists to be included with appropriate  
2485 namespace qualification.

2486 **Example 7-40: Usage of alternative Code Lists**

```
2487 <xsd:complexType name="PersonPropertyCodeType">
2488 <xsd:annotation>
2489 ... see annotation ...
2490 </xsd:annotation>
2491 <xsd:choice>
2492 <xsd:element ref="clm63479:MaritalCode"/>
2493 <xsd:element ref="clm63499:GenderCode"/>
2494 </xsd:choice>
2495 </xsd:complexType>
```

2496 **7.6.6 Attribute and Element Declarations**

2497 There will be no element declarations in the qualified data type schema module. Attribute declarations in the  
2498 qualified data type schema will either be those present in the base data type with further restrictions applied  
2499 as required, or represented as XSD built-in data type facets such as those conveyed in the namespace  
2500 declaration for code and identifier lists or representing further restrictions to **xsd:dateTime**.



---

[R160] The qualified data type `xsd:complexType` definition `xsd:simpleContent` element MUST only restrict attributes declared in its base type, or MUST only restrict facets equivalent to inherited supplementary components.

---

#### Example 7-41: Qualified Data Type Restricting an Identification Scheme

```
<xsd:complexType name="PartyIDType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:restriction base="udt:IDType">
      <xsd:attribute name="schemeName" use="prohibited"/>
      <xsd:attribute name="schemeAgencyName" use="prohibited"/>
      <xsd:attribute name="schemeVersionID" use="prohibited"/>
      <xsd:attribute name="schemeDataURI" use="prohibited"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
```

[Note] - In the case of decoupling a particular qualified data type from a set of value enumerations, the supplementary component attribute declarations MAY have a default= value for guidance. However, doing so obligates users to explicitly override the value if it does not apply when used.

### 7.6.7 Annotation

---

[R161] Every qualified data type definition MAY contain a structured set of annotations in the following sequence and pattern:

- UniqueID (required): The identifier that references a Qualified Data Type instance in a unique and unambiguous way.
- Acronym (required): The abbreviation of the type of component. In this case the value will always be QDT.
- DictionaryEntryName (required): The official name of the Qualified Data Type.
- Version (required): An indication of the evolution over time of the Qualified Data Type instance.
- Definition (required): The semantic meaning of the Qualified Data Type.
- PrimaryRepresentationTerm (required): The Primary Representation Term of the Qualified Data Type.
- DataTypeQualifierTerm (required): A term that qualifies the Representation Term in order to differentiate it from its underlying Unqualified Data Type and other Qualified Data Type.
- PrimitiveType (required): The primitive data type of the Qualified Data Type.
- BusinessProcessContextValue (optional, repetitive): The business process context for this Qualified Data Type is associated.
- GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for this Qualified Data Type.
- OfficialConstraintContextValue (optional, repetitive): The official constraint context for this Qualified Data Type.
- ProductContextValue (optional, repetitive): The product context for this Qualified Data Type.
- IndustryContextValue (optional, repetitive): The industry context for this Qualified Data Type.
- BusinessProcessRoleContextValue (optional, repetitive): The role context for this Qualified Data Type.
- SupportingRoleContextValue (optional, repetitive): The supporting role context for this Qualified Data Type.
- SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for this Qualified Data Type.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the Qualified Data Type.
- Example (optional, repetitive): Example of a possible value of a Qualified Data Type.

---

#### Example 7-42: Annotation of qualified data types

```
... see type definition ...
<xsd:annotation>
```

```

2559 <xsd:documentation xml:lang="en">
2560 <ccts:UniqueID>UN02000133</ccts:UniqueID>
2561 <ccts:Acronym>QDT</ccts:Acronym>
2562 <ccts:DictionaryEntryName>Subject_ Code. Type</ccts:DictionaryEntryName>
2563 <ccts:Version>1.0</ccts:Version>
2564 <ccts:Definition>A character string used to represent a subject code.
2565 </ccts:Definition>
2566 <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>
2567 <ccts:PrimitiveType>string</ccts:PrimitiveType>
2568 <ccts:DataTypeQualifierTerm>Subject</ccts:DataTypeQualifierTerm>
2569 </xsd:documentation>
2570 ... see type definition ...
2571

```

---

[R162] For every supplementary component **xsd:attribute** declaration a structured set of annotations MAY be present in the following pattern:

- UniqueID (optional): The identifier that references a Supplementary Component of a Core Component Type instance in a unique and unambiguous way.
  - Acronym (required): The abbreviation of the type of component. In this case the value will always be SC.
  - DictionaryEntryName (required): The official name of a Supplementary Component.
  - Definition (required): The semantic meaning of a Supplementary Component.
  - Cardinality (required): Indication whether the Supplementary Component Property represents a not-applicable, optional, required and/or repetitive characteristic of the Core Component Type.
  - ObjectClassTerm (required): The Object Class Term of the associated Supplementary Component.
  - PropertyTerm (required): The Property Term of the associated Supplementary Component.
  - PrimaryRepresentationTerm (required): The Primary Representation Term of the associated Supplementary Component.
  - PrimitiveType (required): The Primitive Type of the associated Supplementary Component.
  - UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the Supplementary Component.
- 

## 2591 7.7 Code Lists

2592 Codes are an integral component of any business to business information flow. Codes have been developed  
2593 over time to facilitate the flow of compressed, standardized values that can be easily validated for correctness  
2594 to ensure consistent data. In order for the XML instance documents to be fully validated by the parsers, any  
2595 codes used within the XML document need to be available as part of the schema validation process. Many  
2596 international, national and sectoral agencies create and maintain code lists relevant to their area. If required to  
2597 be used within an information flow, these code lists will be stored in their own schema, and are referred to as  
2598 external code lists. For example, many of the existing code lists that exist in the United Nations Code List  
2599 (UNCL) will be stored as external code list schema for use within other UN/CEFACT XSD Schema.

2600 It should be noted that the use of enumerated code lists in messages is entirely optional. Great care should  
2601 be taken when using them due to the issues in maintenance that can be created by their use.

---

2602 [R163] Each UN/CEFACT maintained code list MUST be defined in its own schema module.

---

2603 External code lists must be used when they exist in schema module form and when they can be directly  
2604 imported into a schema module.

2605 UN/CEFACT may design and use an internal code list schema where an existing external code list schema  
2606 needs to be extended, or where no suitable external code list schema exists. If a code list schema is created,  
2607 it should be globally scoped and designed for reuse and sharing.

---

2608 [R164] Internal code list schema MUST NOT duplicate existing external code list schema when the  
2609 existing ones are available to be imported.

---

2610 **7.7.1 Schema Construct**

2611 The code list schema module will follow the general pattern for all UN/CEFACT XSD schema modules.  
2612 Following the generic module information, the body of the schema will consist of code list definitions of the  
2613 following general form:  
2614

2615 **Example 7-43: Structure of code lists**

```

2616 <?xml version="1.0" encoding="UTF-8"?>
2617 <!-- ===== -->
2618 <!-- ===== 6Recommendation20 - Code List Schema Module ===== -->
2619 <!-- ===== -->
2620 <!--
2621 Schema agency: UN/CEFACT
2622 Schema version: 2.0
2623 Schema date: [SCHEMADATE]

2624 Code list name: Measurement Unit Common Code
2625 Code list agency: UNECE
2626 Code list version: 3

2627 ... see intellectual property disclaimer ...

2628 -->
2629 <xsd:schema targetNamespace=" ... see namespace ...
2630 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2631 elementFormDefault="qualified" attributeFormDefault="unqualified">
2632 <!-- ===== -->
2633 <!-- ===== Root Element ===== -->
2634 <!-- ===== -->
2635 ... see root element declaration ...
2636 <!-- ===== -->
2637 <!-- ===== Type Definitions ===== -->
2638 <!-- ===== -->
2639 <!-- ===== Type Definition: Measurement Unit Common Code Content Type == -->
2640 <!-- ===== -->
2641 ... see type definition ...
2642 </xsd:schema>

```

2643 **7.7.2 Namespace Name for Code Lists**

2644 The namespace name for code list is unique in order to convey some of the supplementary component  
2645 information rather than including them as attributes. Specifically, the UN/CEFACT namespace structure for a  
2646 namespace name of a code list should be:

2647 **urn:un:unece:uncefact:codelist:<status>:<Code List Agency Identifier|Code List**  
2648 **Agency Name Text>:<Code List Identification Identifier|Code List Name**  
2649 **Text>:<Code List Version Identifier>**

2650 Where:

- 2651 ○ Namespace Identifier (NID) = un
- 2652 ○ Namespace Specific String = uncece:uncefact:codelist:
- 2653 ○ <status> with uncece and uncefact as fixed value second and third level domains within the NID of un  
2654 and the code list as a fixed schema type.
- 2655 ○ Supplementary Component String for unique identifying of code lists = <Code List. Agency  
2656 Identifier|Code List. Agency Name. Text>:<Code List. Identification. Identifier|Code List. Name.  
2657 Text>:<Code List. Version. Identifier>

2658 [R165] The namespace names for code list schemas MUST have the following structure:

2659 **urn:un:unece:uncefact:codelist:<status>:<Code List Agency**  
2660 **Identifier|Code List Agency Name Text>:<Code List Identification.**  
2661 **Identifier|Code List Name Text>:<Code List Version. Identifier>**

2662 Where:

- 2663 codelist = this token identifying the schema as a code list
- 2664 status = a token identifying the standards status of this code list: draft|standard
- 2665 Code List Agency Identifier = identifies the agency that manages a code list. The default  
2666 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.
- 2667 Code List Agency Name Text = the name of the agency that maintains the code list.
- 2668 Code List Identification Identifier = identifies a list of the respective corresponding codes. listID  
2669 is only unique within the agency that manages this code list. Code List Name Text = the  
2670 name of a list of codes.

2671 Code List Version Identifier = identifies the version of a code list.

2672 [R166] This rule was combined with [R165].

---

2673

2674 **Example 7-44: Namespace name of a code list with an agency and a code list identifier at draft**  
2675 **status**

```
2676 "urn:un:unece:unefact:codelist:draft:6:3403:D.04A"  
2677 where  
2678 6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2679 the Code List. Agency. Identifier  
2680 3403 = UN/CEFACT data element tag for Name type code representing the  
2681 Code List. Identification. Identifier  
2682 D.04A = the version of the UN/CEFACT directory
```

2683 **Example 7-45: Namespace name of proprietary code list at draft status**

```
2684 "urn:un:unece:unefact:codelist:draft:Security_Initiative:Document_Security:1.2"  
2685 where  
2686 SecurityInitiative = the code list agency name of a responsible agency, which  
2687 is not defined in UN/CEFACT data element 3055  
2688 representing the Code List. Agency. Identifier  
2689 DocumentSecurity = the value for Code List. Name. Text  
2690 1.2 = the value for Code List. Version. Identifier
```

2691 **Example 7-46: Namespace name of a code list with an agency and a code list identifier at**  
2692 **standard status**

```
2693 "urn:un:unece:unefact:codelist:standard:6:3403:D.04A"  
2694 where  
2695 6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2696 the Code List. Agency. Identifier  
2697 3403 = UN/CEFACT data element tag for Name status code representing the  
2698 Code List. Identification. Identifier  
2699 D.04A = the version of the UN/CEFACT directory
```

2700 **Example 7-47: Namespace name of proprietary code list at standard status**

```
2701 "urn:un:unece:unefact:codelist:standard:Security_Initiative:Document_Security:1.2"  
2702 where  
2703 SecurityInitiative = the code list agency name of a responsible agency, which  
2704 is not defined in UN/CEFACT data element 3055  
2705 representing the Code List. Agency. Identifier  
2706 DocumentSecurity = the value for Code List. Name. Text  
2707 1.2 = the value for Code List. Version. Identifier
```

2708 Versioning for code lists published by external organisations is outside of the control of UN/CEFACT. As with  
2709 UN/CEFACT published code lists and identifier list schema the value of the Code List Version Identifier will  
2710 follow the same rules as for versioning of other schema modules.

### 2711 7.7.3 UN/CEFACT XSD Schema Namespace Token for Code Lists

2712 A unique token will be defined for each namespace of code lists. The token representing the namespace for  
2713 code lists should be constructed based on the identifier of the agency maintaining the code list and the  
2714 identifier of the specific code list as issued by the maintenance agency except where there is no identifier.  
2715 When there is no identifier, the name for the agency and/or code list should be used instead. This will typically  
2716 be true when proprietary code lists are used. This method of token construction will provide uniqueness with  
2717 a reasonably short token. When the code list is used for a qualified data type with a restricted set of valid  
2718 code values, the qualified data type name is required to be used to distinguish one set of restricted values  
2719 from another.

2720 The agency maintaining the code list will generally be either identified by the agency code as specified in data  
2721 element 3055 in the UN/CEFACT Code List directory or the agency name if the agency does not have a code  
2722 value in 3055. The identifier of the specific code list will generally be the data element tag of the  
2723 corresponding list in the UN/CEFACT directory. If there is no corresponding data element, then the name of  
2724 the code list will be used.

2725 In cases where the code list schema is a restricted set of values of a published code list schema, the code list  
2726 schema will be associated with a qualified data type, and the name of the qualified data type will be included  
2727 as part of the namespace token to ensure uniqueness from the unrestricted code list schema.

2728 [R167] Each UN/CEFACT maintained code list schema module MUST be represented by a unique token  
2729 constructed as follows:

```
2730     clm[Qualified data type name]<Code List Agency Identifier|Code List  
2731     Agency Name Text><Code List Identification Identifier|Code List Name  
2732     Text>
```

2733 with any repeated words eliminated.

2734  
2735 **Example 7-48: Code list token with an agency and a code list identifier**

```
2736     The code list token for Name Type. Code is clm63403  
2737     where  
2738     6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2739     the Code List. Agency. Identifier  
2740     3403 = UN/CEFACT data element tag for Name status code representing  
2741     the Code List. Identification. Identifier
```

2742 **Example 7-49: Code list token for a qualified data type with an agency and code list identifiers**

```
2743     Code list token for Person_Name Type. Code is clmPersonNameType63403  
2744     where  
2745     PersonNameType = name of the qualified data type  
2746     6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2747     the Code List. Agency. Identifier  
2748     3403 = UN/CEFACT data element tag for Name status code representing  
2749     the Code List. Identification. Identifier
```

2750 **Example 7-50: Code list token for a proprietary code list**

```
2751     Code list token for a proprietary code list for Document Security is  
2752     clmSecurityInitiativeDocumentSecurity  
2753     where  
2754     SecurityInitiative = the code list agency name of a responsible agency, which  
2755     is not defined in UN/CEFACT data element 3055  
2756     representing the Code List. Agency. Identifier  
2757     DocumentSecurity = the value for Code List. Name. Text
```

2758 Based on the constructs identified in the above examples, a namespace declaration for a code list would  
2759 appear as shown in Example 7-51.

2760 **Example 7-51: Target namespace declaration for a code list**

```
2761     <xsd:schema  
2762     targetNamespace="urn:un:unece:unefact:codelist:draft:6:4437:D.04A"  
2763     xmlns:clm64437="urn:un:unece:unefact:codelist:draft:6:4437:D.04A"  
2764     xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2765     elementFormDefault="qualified" attributeFormDefault="unqualified">
```

2766 [Note]  
2767 External developers are encouraged to follow the above construct rule when customizing  
2768 schema for code lists to ensure that there is no namespace conflict.

2769 **7.7.4 Schema Location**

2770 Schema locations of code lists are typically defined as URL based URI schemes because of resolvability  
2771 limitations of URN based URI schemes. However, UN/CEFACT XSD Schema of code lists use a URN  
2772 based URI scheme for namespace declarations because persistence is considered more important than  
2773 resolvability. In recognition of the need for resolvability of schema location, until such time as URNs become  
2774 fully resolvable, UN/CEFACT will store schema of code lists in locations identified using a URL based URI  
2775 scheme aligned with the URN based URI scheme used for the namespace declaration as follows:

```
2776     urn:un:unece:unefact:codelist:<status>:<Code List. Agency Identifier|Code  
2777     List. Agency Name. Text>:<Code List. Identification. Identifier|Code List.  
2778     Name. Text>:<Code List. Version. Identifier>
```

- 
- 2779 [R168] The structure for schema location of code lists MUST be:
- 2780 `../codelist/<status>/<Code List Agency Identifier|Code List Agency`  
 2781 `Name Text>/<Code List Identification Identifier|Code List Name`  
 2782 `Text>_<Code List Version Identifier>.xsd`
- 2783 Where:
- 2784 `schematype` = a token identifying the type of schema module: `codelist`  
 2785 `status` = the status of the schema as: `draft|standard`  
 2786 `Code List Agency Identifier` = identifies the agency that manages a code list. The default  
 2787 agencies used are those from DE 3055. `Code List Agency Name Text` = the name of the  
 2788 agency that maintains the code list.  
 2789 `Code List Identification Identifier` = identifies a list of the respective corresponding codes. `listID` is  
 2790 only unique within the agency that manages this code list.  
 2791 `Code List Name Text` = the name of a list of codes.  
 2792 `Code List Version Identifier` = identifies the version of a code list.
- 2793 [R169] Each `xsd:schemaLocation` attribute declaration of a code list MUST contain a resolvable URL,  
 2794 and if an absolute path is used, it MUST also be persistent.
- 2795 [R170] This rule has been removed.
- 

## 2796 7.7.5 Imports and Includes

2797 UN/CEFACT Code List Schema Modules are standalone schema modules and will not import or include any  
 2798 other schema modules.

- 
- 2799 [R171] Code List schema modules MUST not import or include any other schema modules.
- 

## 2800 7.7.6 Type Definitions

2801 [R172] Within each code list module one, and only one, named `xsd:simpleType` MUST be defined for  
 2802 the content component.

2803 [R173] The name of the `xsd:simpleType` MUST be the name of code list root element with the word  
 2804 'ContentType' appended.

---

### 2805 Example 7-52: Simple type definition of code lists

```

2806 <!-- ===== Type Definitions ===== -->
2807 <!-- ===== Type Definitions ===== -->
2808 <!-- ===== Type Definitions ===== -->
2809 <!-- ===== Type Definition: Party Role Type Code ===== -->
2810 <!-- ===== Type Definition: Party Role Type Code ===== -->
2811 <xsd:simpleType name="PartyRoleCodeContentType">
2812   <xsd:restriction base="xsd:token">
2813     <xsd:enumeration value="AA">
2814       ... see enumeration ...
2815     </xsd:enumeration>
2816     ...
2817   </xsd:restriction>
2818 </xsd:simpleType>
  
```

2819 [R174] The `xsd:restriction` element base attribute value MUST be set to `xsd:token`.

2820 [R175] Each code in the code list MUST be expressed as an `xsd:enumeration`, where the  
 2821 `xsd:value` for the enumeration is the actual code value.

---

### 2822 Example 7-53: Enumeration facet of code lists

```

2823 ... see type definition ...
2824 <xsd:enumeration value="AA">
2825   <xsd:annotation>
2826     ... see annotation
2827   </xsd:annotation>
2828 </xsd:enumeration>
2829 <xsd:enumeration value="AB">
2830   <xsd:annotation>
  
```



2831  
2832  
2833  
2834

```
... see annotation
</xsd:annotation>
</xsd:enumeration>
...
```

2835 The purpose of the code list schema module is to define the list of allowable values (enumerations) that can  
2836 appear within a particular element. Facet restrictions may be included in code lists if desired for additional  
2837 edit check validation.

### 2838 7.7.7 Element and Attribute Declarations

2839 Each code list schema module will have a single `xsd:simpleType` defined. This single `xsd:simpleType`  
2840 definition will have a `xsd:restriction` expression whose base is a XSD built-in data type. The  
2841 `xsd:restriction` will be used to convey the content component enumeration value(s).

---

2842 [R176] For each code list a single root element MUST be globally declared.

2843 [R177] The name of the code list root element MUST be the name of the code list following the  
2844 naming rules as defined in section 5.3.

---

2845 [R178] The code list root element MUST be of a type representing the actual list of code values.

---

#### 2846 Example 7-54: Root element declaration of code lists

2847  
2848  
2849  
2850

```
<!-- ===== -->
<!-- ===== Root Element ===== -->
<!-- ===== -->
<xsd:element name="PartyRoleCode" type="clm63035:PartyRoleCodeContentType"/>
```

2851 The global declaration of a root element for each code list allows the use of code lists from different  
2852 namespaces in a schema module when using `xsd:choice`.

#### 2853 Example 7-55: Usage of a choice of code lists

2854  
2855  
2856  
2857  
2858  
2859  
2860  
2861  
2862

```
<xsd:complexType name="CalculationCurrencyCode">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:choice>
    <xsd:element ref="clm54217-N:CurrencyCode"/>
    <xsd:element ref="clm54217-A:CurrencyCode"/>
  </xsd:choice>
</xsd:complexType>
```

### 2863 7.7.8 Extension and Restriction

2864 Users of the UN/CEFACT library may identify any subset or superset they wish from a specific code list for  
2865 their own trading community requirements by defining a qualified data type.

2866 Representation of a qualified data type of code lists could be

- 2867 ○ a combination of several individual code lists using `xsd:union`
- 2868 ○ a choice between several code lists, using `xsd:choice`
- 2869 ○ subsetting an existing code list using `xsd:restriction`

2870 Each of these can easily be accommodated in this syntax solution as required by the user's business  
2871 requirements. Appendix D provides detailed examples of the various code list options.

### 2872 7.7.9 Annotation

2873 In order to facilitate a clear and unambiguous understanding of the list of allowable codes within an  
2874 element, annotations will be provided for each enumeration to provide the code name and description.

---

2875 [R179] Each code list `xsd:enumeration` MAY contain a structured set of annotations in the  
2876 following sequence and pattern:

- 2877 ○ Name (required): The name of the code.
  - 2878 ○ Description (optional): Descriptive information concerning the code.
- 

2879

2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888

### Example 7-56: Annotation of codes

```
<xsd:enumeration value="AI">  
  <xsd:annotation>  
    <xsd:documentation xml:lang="en">  
      <cts:Name>Successful job applicant</cts:Name>  
      <cts:Description>Person who has been chosen for a job. </cts:Description>  
    </xsd:documentation>  
  </xsd:annotation>  
</xsd:enumeration>...
```

2889

## 7.8 Identifier List Schema

2890  
2891  
2892  
2893

When required, separate schema modules will be defined for identification schemes that have a token, and optionally a description, and that have the same functionality as a code list. In this way, XML instance documents containing these identifiers can be fully validated by the parsers. Other identifier schemes should be defined as a qualified or unqualified data type as appropriate.

2894  
2895

External identifier lists must be used when they exist in schema module form and when they can be directly imported into a schema module.

2896  
2897  
2898

UN/CEFACT may design and use an internal identifier list where an existing external identifier list needs to be extended, or where no suitable external identifier list exists. If an identifier list is created, the lists should be globally scoped and designed for reuse and sharing.

2899  
2900

It should be noted that the use of enumerated identifier lists in messages is entirely optional. Great care should be taken when using them due to the issues in maintenance that can be created by their use.

2901  
2902

---

[R180] Internal identifier lists schema MUST NOT duplicate existing external identifier list schema when the existing ones are available to be imported.

2903

[R181] Each UN/CEFACT maintained identifier list MUST be defined in its own schema module.

---

2904

### 7.8.1 Schema Construct

2905  
2906  
2907

The identifier list schema module will follow the general pattern for all UN/CEFACT XSD schema modules. Following the generic module information, the body of the schema will consist of identifier list definitions of the following general form:

2908

#### Example 7-57: Structure of identifier lists

2909  
2910  
2911  
2912  
2913  
2914  
2915  
2916

```
<?xml version="1.0" encoding="UTF-8"?>  
<!-- ===== -->  
<!-- ===== Agency Identifier - Identifier List Schema Module ===== -->  
<!-- ===== -->  
<!--  
  Schema agency:      UN/CEFACT  
  Schema version:    2.0  
  Schema date:       [SCHEMADATE]
```

2917  
2918  
2919

```
  Identifier list name:      Agency Identifier  
  Identifier list agency:    UNECE  
  Code list version:      3
```

2920

```
  ... see intellectual property disclaimer ...
```

2921

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

2933

## 2934 7.8.2 Namespace Name For Identifier List Schema

2935 The namespace name for identifier list is unique in order to convey some of the supplementary component  
2936 information rather than including them as attributes. Specifically, the UN/CEFACT namespace structure for a  
2937 namespace name of an identifier list schema should be:

```
2938 urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme Agency  
2939 Identifier|Identifier Scheme Agency Name Text>:< Identifier Scheme  
2940 Identifier|Identifier Scheme Name Text>:< Identifier Scheme Version.  
2941 Identifier>
```

2942 Where:

- 2943 ○ Namespace Identifier (NID) = un
- 2944 ○ Namespace Specific String =
- 2945 ○ unece:uncefact:odelist:<status> with unece and uncefact as fixed value second and third level  
2946 domains within the NID of un and the code list as a fixed schema type.
- 2947 ○ Supplementary Component String for unique identifying of identifier schemes = <Identifier Scheme  
2948 Agency Identifier|Identifier Scheme Agency Name Text>:< Identifier Scheme Identifier|Identifier  
2949 Scheme Name Text>:< Identifier Scheme Version Identifier>

---

2950 [R182] The names for namespaces MUST have the following structure:

```
2951 urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme.  
2952 Agency Identifier|Identifier Scheme Agency Name Text>:<Identifier  
2953 Scheme Identifier|Identifier Scheme Name Text>:<Identifier Scheme  
2954 Version Identifier>
```

2955 Where:

- 2956 status = the token identifying the publication status of this identifier scheme schema =  
2957 draft|standard
- 2958 identifierlist = this token identifying the schema as an identifier scheme
- 2959 Identifier Scheme Agency Identifier = the identification of the agency that maintains the  
2960 identification scheme.
- 2961 Identifier Scheme Agency Name. Text = the name of the agency that maintains the  
2962 identification list.
- 2963 Identifier Scheme Identifier = the identification of the identification scheme.
- 2964 Identifier Scheme Name. Text = the name of the identification scheme.
- 2965 Identifier Scheme Version. Identifier = the version of the identification scheme.

2966 [R183] This rule was combined with [R182].

---

### 2967 **Example 7-58: Namespace name of an identifier list schema with an agency and an identifier list** 2968 **schema identifier at draft status**

```
2969 "urn:un:unece:uncefact:identifierlist:draft:5:3166:2001"  
2970 where  
2971 5 = the value for ISO in UN/CEFACT data element 3055 representing the Code List.  
2972 Agency. Identifier  
2973 4217 = ISO identifier scheme identifier for country code representing the Code List.  
2974 Identification. Identifier  
2975 2001 = the version of the ISO country identifier list.
```

### 2976 **Example 7-59: Namespace of an identifier list schema with an agency and an identifier list schema** 2977 **identifier at standard status**

```
2978 "urn:un:unece:uncefact:identifierlist:standard:5:3166:2001"  
2979 where  
2980 5 = the value for ISO in UN/CEFACT data element 3055 representing the Code List.  
2981 Agency. Identifier  
2982 4217 = ISO identifier scheme identifier for country code representing the Code List.  
2983 Identification. Identifier  
2984 2001 = the version of the ISO country identifier list.
```

2985 Versioning for identifier list schemas published by external organisations is outside of the control of  
2986 UN/CEFACT. As with UN/CEFACT published identifier list schema the value of the Identifier Scheme Version  
2987 Identifier will follow the same rules as for versioning of other schema modules.

### 7.8.3 UN/CEFACT XSD Namespace Token for Identifier List Schema

A unique token will be defined for each namespace of an identifier list schema. The token representing the namespace for identifier lists should be constructed based on the identifier of the agency maintaining the identification list and the identifier of the specific identification list as issued by the maintenance agency. This method of token construction will provide uniqueness with a reasonably short token. When the identifier list is used for a qualified data type with a restricted set of valid identifier values, the qualified data type name is required to be used to distinguish one set of restricted values from another.

The agency maintaining the identification list will be either identified by the agency code as specified in data element 3055 in the UN/CEFACT EDIFACT directory. The identifier of the identification list will be the identifier as allocated by the identification scheme agency.

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

---

[R184] Each UN/CEFACT maintained identifier list schema module MUST be represented by a unique token constructed as follows:

```
ids[Qualified data type name]<Identification Scheme Agency  
Identifier><Identification Scheme Identifier>
```

with any repeated words eliminated.

---

#### Example 7-60: Identifier list token

```
Token for the ISO Country Codes would be: ids53166-1  
where:  
5 = the Identification Scheme Agency Identifier for ISO in codelist 3055 3166-1 = the  
Identification Scheme Identifier as allocated by ISO.
```

Based on the constructs identified in Example 7-60, a namespace declaration for an identifier list would appear as shown in Example 7-61.

#### Example 7-61: Target Namespace declaration for an Identifier list

```
<xsd:schema  
targetNamespace="urn:un:unece:uncefact:identifierlist:draft:5:3166-1:1997"  
xmlns:ids53166-1="urn:un:unece:uncefact:identifierlist:draft:5:3166-1:1977"  
xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

[Note]

External developers are encouraged to follow the above construct rule when customizing schema for identifier lists to ensure that there is no namespace conflict.

### 7.8.4 Schema Location

Schema locations of identifier list schema are typically defined as URL based URI schemes because of resolvability limitations of URN based URI schemes. However, UN/CEFACT XSD Schema of identifier lists use a URN based URI scheme for namespace declarations because persistence is considered more important than resolvability. In recognition of the need for resolvability of schema location, until such time as URNs become fully resolvable, UN/CEFACT will store schema of identifier list in locations identified using a URL based URI scheme aligned with the URN based URI scheme used for the namespace declaration as follows:

```
urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme Agency  
Identifier|Identifier Scheme Agency Name Text>:< Identifier Scheme  
Identifier|Identifier Scheme Name Text>:< Identifier Scheme Version.  
Identifier>
```

---

[R185] The structure for schema location of identifier lists MUST be:

```
[./identifierlist/<status>/<Identifier Scheme Agency Identifier|Identifier  
Scheme Agency Name Text>/< Identifier Scheme Identifier|Identifier  
Scheme Name Text>_< Identifier Scheme Version Identifier>.xsd
```

3038 Where:  
3039 schematype = a token identifying the type of schema module: `identifierlist`  
3040 status = the status of the schema as: `draft|standard`  
3041 Identifier Scheme. Agency Identifier = the identification of the agency that maintains the  
3042 identification scheme.  
3043 Identifier Scheme. Agency Name. Text = the name of the agency that maintains the  
3044 identification scheme.  
3045 Identifier Scheme. Identifier = the identification of the identification scheme.  
3046 Identifier Scheme. Name. Text = the name of the identification scheme.  
3047 Identifier Scheme. Version. Identifier = the version of the identification scheme.

3048 [R186] Each `xsd:schemaLocation` attribute declaration of an identifier list schema MUST contain a  
3049 resolvable URL, and if an absolute path is used, it MUST also be persistent.

3050 [R187] This rule has been removed.

---

## 3051 7.8.5 Imports and Includes

3052 UN/CEFACT Identifier List Schema Modules are standalone schema modules and will not import or include  
3053 any other schema modules.

3054 [R188] Identifier list schema modules MUST NOT import or include any other schema modules.

---

## 3055 7.8.6 Type Definitions

3056 A restriction has to be declared in order to define the content component (the simple type) as a restriction of  
3057 the unqualified data type in order to comply with parser requirements. The restriction itself is the list of  
3058 enumerations.

3059 [R189] Within each identifier list schema module one, and only one, named `xsd:simpleType` MUST  
3060 be defined for the content component.

3061 [R190] The name of the `xsd:simpleType` MUST be the name of the identifier list root element with the  
3062 word 'ContentType' appended.

---

### 3063 Example 7-62: Simple type definition of an identifier list

```
3064 <!-- ===== Type Definitions ===== -->  
3065 <!-- ===== -->  
3066 <xsd:simpleType name="PaymentTermsDescriptionIdentifierContentType">  
3067 <xsd:restriction base="xsd:token">  
3068 <xsd:enumeration value="1">  
3069 ... see enumeration ...  
3070 </xsd:enumeration>  
3071 </xsd:restriction>  
3072 </xsd:simpleType>
```

---

3073 [R191] The `xsd:restriction` element base attribute value MUST be set to `xsd:token`.

3074 [R192] Each identifier in the identifier list MUST be expressed as an `xsd:enumeration`, where the  
3075 `xsd:value` for the enumeration is the actual identifier value.

---

### 3076 Example 7-63: Enumeration facet of an identifier list

```
3077 ... see type definition ...  
3078 <xsd:enumeration value="1">  
3079 <xsd:annotation>  
3080 ... see annotation  
3081 </xsd:annotation>  
3082 </xsd:enumeration>  
3083 <xsd:enumeration value="2">  
3084 <xsd:annotation>  
3085 ... see annotation  
3086 </xsd:annotation>  
3087 </xsd:enumeration>  
3088 ...
```

3089 The purpose of the identifier list schema module is to define the list of allowable values (enumerations) that  
3090 can appear within a particular element. Therefore, no other facet restrictions are allowed.

3091 

---

**[R193]** Facets other than `xsd:enumeration` MUST NOT be used in the identifier list schema module.

---

## 3092 7.8.7 Attribute and Element Declarations

3093 Each identifier list schema module will have a single `xsd:simpleType` defined. This single  
3094 `xsd:simpleType` definition will have a `xsd:restriction` expression whose base is a XSD built-in data  
3095 type. The `xsd:restriction` will be used to convey the content component enumeration value(s).

3096 

---

**[R194]** For each identifier list a single root element MUST be globally declared.

---

3097 **[R195]** The name of the identifier list root element MUST be the name of the identifier list following the  
3098 naming rules as defined in section 5.3.

3099 

---

**[R196]** The identifier list root element MUST be of a type representing the actual list of identifier values.

---

### 3100 Example 7-64: Root element declaration of identifier lists

```
3101 <!-- ===== -->  
3102 <!-- ===== Root Element ===== -->  
3103 <!-- ===== -->  
3104 <xsd:element name="PaymentTermsDescriptionIdentifier" type="ids64277":  
3105 PaymentTermsDescriptionIdentifierContentType"/>
```

3106 The global declaration of a root element for each identifier list allows the use of identifier lists from different  
3107 namespaces in a schema module when using `xsd:choice`.

### 3108 Example 7-65: Usage of a choice of identifier lists

```
3109 <xsd:complexType name="CalculationCurrencyCode">  
3110 <xsd:annotation>  
3111 ... see annotation ...  
3112 </xsd:annotation>  
3113 <xsd:choice>  
3114 <xsd:element ref="clm54217-N:CurrencyCode"/>  
3115 <xsd:element ref="clm54217-A:CurrencyCode"/>  
3116 </xsd:choice>  
3117 </xsd:complexType>
```

## 3118 7.8.8 Extension and Restriction

3119 Users of the UN/CEFACT library may identify any subset or superset they wish from a specific identifier list for  
3120 their own trading community requirements by defining a qualified data type.

3121 Representation of a qualified data type of identifier lists could be

- 3122 ○ a combination of several individual identifier lists using `xsd:union`
- 3123 ○ a choice between several identifier lists, using `xsd:choice`
- 3124 ○ subsetting an existing code list using `xsd:restriction`

3125 Each of these can easily be accommodated in this syntax solution as required by the user's business  
3126 requirements. Appendix D provides detailed examples of the various identifier list options.

3127 XML declarations for using identifier lists in qualified data types are shown in the following examples.

### 3128 Example 7-66: Enumeration facet of identifier scheme

```
3129 ... see type definition ...  
3130 <xsd:enumeration value="AD">  
3131 <xsd:annotation>  
3132 ... see annotation ...  
3133 </xsd:annotation>  
3134 </xsd:enumeration>  
3135 <xsd:enumeration value="AE">  
3136 <xsd:annotation>  
3137 ... see annotation ...  
3138 </xsd:annotation>  
3139 </xsd:enumeration>  
3140 <xsd:enumeration value="AF">
```

3141  
3142  
3143  
3144

```
<xsd:annotation>  
  ... see annotation ...  
</xsd:annotation>  
</xsd:enumeration>
```

3145 **Example 7-67: Usage of only one identifier scheme**

3146  
3147  
3148  
3149  
3150  
3151

```
<xsd:simpleType name="CountryIDType">  
  <xsd:annotation>  
    ... see annotation ...  
  </xsd:annotation>  
  <xsd:restriction base="ids53166:CountryIDContentType"/>  
</xsd:simpleType>
```

3152 **Example 7-68: Usage of alternative identifier schemes**

3153  
3154  
3155  
3156  
3157  
3158  
3159  
3160  
3161

```
<xsd:complexType name="GeopoliticalIDType">  
  <xsd:annotation>  
    ... see annotation ...  
  </xsd:annotation>  
  <xsd:choice>  
    <xsd:element ref="ids53166:CountryCode"/>  
    <xsd:element ref="ids53166-2:RegionCode"/>  
  </xsd:choice>  
</xsd:complexType>
```

3162 **7.8.9 Annotation**

3163 In order to facilitate a clear and unambiguous understanding of the list of allowable identifiers within an  
3164 element, annotations will be provided for each enumeration to provide the name, and optionally a description  
3165 of, the identifier.

---

3166 [R197] Each `xsd:enumeration` MAY contain a structured set of annotations in the following sequence  
3167 and pattern:

- 3168 ○ Name (required): The name of the identifier.
  - 3169 ○ Description (optional): Descriptive information concerning the identifier.
- 

3170 **Example 7-69: Annotation of Identifiers**

3171  
3172  
3173  
3174  
3175  
3176  
3177  
3178  
3179

```
<xsd:enumeration value="1">  
  <xsd:annotation>  
    <xsd:documentation xml:lang="en">  
      <ccts:Name>Draft(s) drawn on issuing bank</ccts:Name>  
      <ccts:Description>Draft(s) must be drawn on the issuing  
        bank.</ccts:Description>  
    </xsd:documentation>  
  </xsd:annotation>  
</xsd:enumeration>
```



## 3180 8 XML Instance Documents

3181 In order to be UN/CEFACT conformant, an instance document must be valid against the relevant  
3182 UN/CEFACT compliant XML schema. The XML instance documents should be readable and understandable  
3183 by both humans and applications, and should enable reasonably intuitive interactions. It should represent all  
3184 truncated tag names as described in section 7. A XPath navigation path should describe the complete  
3185 semantic understanding by concatenating the nested elements. This navigation path should also reflect the  
3186 meaning of each dictionary entry name of a BBIE or ASBIE.

### 3187 8.1 Character Encoding

3188 In conformance with ISO/IEC/ITU/UNECE Memorandum of Understanding Management Group (MOUMG)  
3189 Resolution 01/08 (MOU/MG01n83) all UN/CEFACT XML will be instantiated using UTF. UTF-8 is the  
3190 preferred encoding, but UTF-16 may be used where necessary to support other languages.

---

3191 [R198] All UN/CEFACT XML MUST be instantiated using UTF. UTF-8 should be used as the preferred  
3192 encoding. If UTF-8 is not used, UTF-16 MUST be used.

---

### 3193 8.2 xsi:schemaLocation

3194 The `xsi:schemaLocation` and `xsi:noNamespaceLocation` attributes are part of the XML schema  
3195 instance namespace (<http://www.w3.org/2001/XMLSchema-instance>). To ensure consistency, the token `xsi`  
3196 will be used to represent the XML schema instance namespace.

---

3197 [R199] The `xsi` prefix MUST be used where appropriate for referencing `xsd:schemaLocation` and  
3198 `xsd:noNamespaceLocation` attributes in instance documents.

---

### 3199 8.3 Empty Content

3200 Empty elements do not provide the level of assurance necessary for business information exchanges and as  
3201 such, will not be used.

---

3202 [R200] UN/CEFACT conformant instance documents MUST NOT contain an element devoid of content.

3203 [R201] The `xsi:nil` attribute MUST NOT appear in any conforming instance.

---

### 3204 8.4 xsi:type

3205 The `xsi:type` attribute allows for substitution during an instantiation of a xml document. In the same way  
3206 that substitution groups are not allowed, the `xsi:type` attribute is not allowed.

---

3207 [R202] The `xsi:type` attribute MUST NOT be used

---

3208

3209  
3210  
3211  
3212  
3213  
3214

## Appendix A Related Documents

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

- UN/CEFACT Core Components Technical Specification, Part 8 of the ebXML Framework Version 2.01 with corrigenda.
- UN/CEFACT Core Components Business Document Assembly Technical Specification.

3215  
3216  
3217

## Appendix B Overall Structure

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

- 3218 • XML Declaration
- 3219 • Schema Module Identification and Copyright Information
- 3220 • Schema Start-Tag
- 3221 • Includes
- 3222 • Imports
- 3223 • Element
- 3224 • Root Element
- 3225 • Global Elements
- 3226 • Type Definitions

### B.1 XML Declaration

3227 A UTF-8 encoding is adopted throughout all UN/CEFACT XML schema, unless characters are required that  
3228 are not in UTF-8, in which case UTF-16 can be used.

#### 3230 Example B-1: XML Declaration

```
3231 <?xml version="1.0" encoding="UTF-8"?>
```

### B.2 Schema Module Identification and Intellectual Property Disclaimer

#### 3234 Example B-2: Schema Module Identification and Intellectual Property Disclaimer

```
3235 <!-- ===== -->  
3236 <!-- ===== Example - Schema Module Name ===== -->  
3237 <!-- ===== -->  
3238 <!--  
3239 Schema agency: UN/CEFACT  
3240 Schema version: 2.0  
3241 Schema date: [SCHEMADATE]
```

```
3242 ECE draws attention to the possibility that the practice or implementation of its  
3243 outputs (which include but are not limited to Recommendations, norms, standards,  
3244 guidelines and technical specifications) may involve the use of a claimed intellectual  
3245 property right.  
3246 Each output is based on the contributions of participants in the UN/CEFACT process, who  
3247 have agreed to waive enforcement of their intellectual property rights pursuant to the  
3248 UN/CEFACT IPR Policy (document ECE/TRADE/C/CEFACT/2010/20/Rev.2 available at  
3249 http://www.unece.org/cefact/cf\_docs.html or from the ECE secretariat). ECE takes no  
3250 position concerning the evidence, validity or applicability of any claimed intellectual  
3251 property right or any other right that might be claimed by any third parties related to  
3252 the implementation of its outputs. ECE makes no representation that it has made any  
3253 investigation or effort to evaluate any such  
3254 rights.  
3255 Implementers of UN/CEFACT outputs are cautioned that any third-party intellectual  
3256 property rights claims related to their use of a UN/CEFACT output will be their  
3257 responsibility and are urged to ensure that their use of UN/CEFACT outputs does not  
3258 infringe on an intellectual property right of a third party.  
3259 ECE does not accept any liability for any possible infringement of a claimed  
3260 intellectual property right or any other right that might be claimed to relate to the  
3261 implementation of any of its outputs.  
3262 -->
```

### B.3 Schema Start Tag

3263 The Schema Start-Tag section of an UN/CEFACT compliant XML schema must contain one or more of the  
3264 below declarations as relevant. Relevant declarations must appear in the order given:  
3265

- 3266 • Version
- 3267 • Namespaces
  - 3268 ○ targetNamespace attribute xmlns:xsd attribute
  - 3269 ○ namespace declaration for current schema
  - 3270 ○ namespace declaration for reusable ABIEs actually used in the schema
  - 3271 ○ namespace declaration for unqualified data types actually used in the schema
  - 3272 ○ namespace declaration for qualified data types actually used in the schema
  - 3273 ○ namespace declaration for code lists actually used in the schema
  - 3274 ○ namespace declaration for identifier schemes actually used in the schema
  - 3275 ○ namespace declaration for CCTS
- 3276 • Form Defaults elementFormDefault attributeFormDefault
- 3277 • Others
  - 3278 ○ other schema attributes with schema namespace
  - 3279 ○ other schema attributes with non-schema namespace

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

```

3281 <xsd:schema
3282 xmlns:ccts="urn:un:unece:uncefact:documentation:standard:CoreComponentsTechnicalSpecificat
3283 ion:2"
3284 xmlns:clm6Recommendation20="urn:un:unece:uncefact:codelist:standard:6:Recommendation20:8"
3285 xmlns:clm60133="urn:un:unece:uncefact:codelist:standard:6:0133:40106"
3286 xmlns:clm5ISO42173A="urn:un:unece:uncefact:codelist:standard:5:ISO42173A:2012-08-31"
3287 xmlns:ids5ISO316612A="urn:un:unece:uncefact:identifierlist:standard:5:ISO316612A:SecondEdi
3288 tion2006VI-13"
3289 xmlns:clmIANAMIMEMediaType="urn:un:unece:uncefact:codelist:standard:IANA:MIMEMediaType:201
3290 3-01-03"
3291 xmlns:clmIANACHaracterSetCode="urn:un:unece:uncefact:codelist:standard:IANA:CharacterSetCo
3292 de:2013-01-08" xmlns:clm63055="urn:un:unece:uncefact:codelist:standard:6:3055:D12A"
3293 xmlns:udt="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:13"
3294 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3295 targetNamespace="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:13"
3296 elementFormDefault="qualified"
3297 attributeFormDefault="unqualified"
3298 version="13.0">

```

## B.4 Includes

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

- Inclusion of the internal ABIE schema module if used

### Example B-4: Includes

```

3304 <!-- ===== -->
3305 <!-- ===== Include ===== -->
3306 <!-- ===== -->
3307 <!-- ===== Inclusion of internal ABIE ===== -->
3308 <!-- ===== -->
3309 <xsd:include
3310 namespace="urn:un:unece:uncefact:data:standard:CIIAggregateBusinessInformationEntity
3311 :1"
3312 schemaLocation="http://www.unece.org/uncefact/data/standard/CIIAggregateBusinessInformationE
3313 ntity_1.xsd"/>

```

## B.5 Imports

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

- Import of the reusable ABIE schema module if used
- Import of the unqualified data type schema module if used
- Import of the qualified data type schema module if used
- Import of code list schema modules actually used
- Import of identifier list schema modules actually used

3322 **Example B-5: Imports**

```

3323 <!-- ===== -->
3324 <!-- ===== Imports ===== -->
3325 <!-- ===== -->
3326 <!-- ===== Import of Unqualified Data Type ===== -->
3327 <!-- ===== -->
3328 <xsd:import namespace="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:14"
3329 schemaLocation="http://www.unece.org/uncefact/data/standard/UnqualifiedDataType_14.xsd" />
3330 <!-- ===== -->
3331 <!-- ===== Import of Qualified Data Type ===== -->
3332 <!-- ===== -->
3333 <xsd:import namespace="urn:un:unece:uncefact:data:standard:QualifiedDataType:14"
3334 schemaLocation="http://www.unece.org/uncefact/data/standard/QualifiedDataType_14.xsd"/
3335 >
3336 <!-- ===== -->
3337 <!-- ===== Import of Reusable Aggregate Business Information Entity Schema Module -->
3338 <!-- ===== -->
3339 <xsd:import
3340 namespace="urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEn
3341 tity:13" schemaLocation="ReusableAggregateBusinessInformationEntity_13p0.xsd"/>

```

3342 **B.6 Root Element**

3343 The root element is declared first when needed in schema that are used to support instance documents.  
3344 Global elements are then declared following the root element when it is present.

3345 **Example B-6:**

```

3346 <!-- ===== -->
3347 <!-- ===== Element Declarations ===== -->
3348 <!-- ===== -->
3349 <!-- ===== Root element ===== -->
3350 <!-- ===== -->
3351 <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]">
3352 <!-- ===== -->
3353 <!-- ===== Global Element Declarations ===== -->
3354 <!-- ===== -->
3355 <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]">
3356 <!-- ===== -->

```

3357 The root element's type definition is defined immediately following the definition of the global root element to  
3358 provide clear visibility of the root element's type, of which this particular schema is all about.

3359 **Example B-7:**

```

3360 <!-- ===== -->
3361 <!-- ===== Root element ===== -->
3362 <!-- ===== -->
3363 <xsd:element name="CrossIndustryInvoice" type="rsm:CrossIndustryInvoiceType">
3364 <xsd:annotation>
3365 <xsd:documentation>
3366 <ccts:UniqueID>CII</ccts:UniqueID>
3367 <ccts:Acronym>RSM</ccts:Acronym>
3368 <ccts:Name>CrossIndustryInvoice</ccts:Name>
3369 <ccts:Version>1.0</ccts:Version>
3370 <ccts:Definition>A message used as a request for payment, or modification of
3371 a request for payment, for the supply of goods or services ordered,
3372 delivered, received, consumed.</ccts:Definition>
3373 <ccts:BusinessProcessContextValue>PurchaseOrder</ccts:BusinessProcessContextValue>
3374 <ccts:GeopoliticalOrRegionContextValue>In All
3375 Contexts</ccts:GeopoliticalOrRegionContextValue>
3376 <ccts:OfficialConstraintContextValue>None</ccts:OfficialConstraintContextValue>
3377 <ccts:ProductContextValue>In All Contexts</ccts:ProductContextValue>
3378 <ccts:IndustryContextValue>In All Contexts</ccts:IndustryContextValue>
3379 <ccts:BusinessProcessRoleContextValue>In All
3380 Contexts</ccts:BusinessProcessRoleContextValue>
3381 <ccts:SupportingRoleContextValue>In All
3382 Contexts</ccts:SupportingRoleContextValue>
3383 <ccts:SystemCapabilitiesContextValue>In All
3384 Contexts</ccts:SystemCapabilitiesContextValue>
3385 </xsd:documentation>
3386 </xsd:annotation>
3387 </xsd:element>

```

3388

**Example B-8: Global elements**

```

3389 <!-- ===== -->
3390 <!-- ===== Global element ===== -->
3391 <!-- ===== -->
3392 <xsd:element name="AdministrativeCountrySubDivision"
3393 type="ram:AdministrativeCountrySubDivisionType"/>
3394 <xsd:annotation>
3395 <xsd:documentation>
3396 <ccts:UniqueID>UN01009362</ccts:UniqueID>
3397 <ccts:Acronym>ABIE</ccts:Acronym>
3398 <ccts:DictionaryEntryName>Administrative_ Country Subdivision.
3399 Details</ccts:DictionaryEntryName>
3400 <ccts:Version>1.0</ccts:Version>
3401 <ccts:Definition>An area which is an administrative sub-division within a
3402 country, such as a state, a county, a canton or a province.</ccts:Definition>
3403 <ccts:ObjectClassTerm>Country Subdivision<ccts:ObjectClassTerm>
3404 <ccts:QualifierTerm>Administrative<ccts:QualifierTerm>
3405 </xsd:documentation>
3406 </xsd:annotation>
3407 </xsd:element>

```

3408

**B.7 Type Definitions**

- 3409 ○ Definition of types for Aggregate Business Information Entities in alphabetical order, if applicable.
- 3410 ○ Definition of types for Basic Business Information Entities in alphabetical order, if applicable.

3411

**Example B-9: Type Definitions**

```

3412 <!-- ===== -->
3413 <!-- ===== Type Definitions ===== -->
3414 <!-- ===== -->
3415 <!-- ===== Type Definition: Administrative SubDivision type ===== -->
3416 <!-- ===== -->
3417 <xsd:complexType name="AdministrativeCountrySubDivisionType">
3418 <xsd:annotation>
3419 <xsd:documentation xml:lang="en">
3420 <ccts:UniqueID>UN01009362</ccts:UniqueID>
3421 <ccts:Acronym>ABIE</ccts:Acronym>
3422 <ccts:DictionaryEntryName>Administrative_ Country Subdivision.
3423 Details</ccts:DictionaryEntryName>
3424 <ccts:Version>1.0</ccts:Version>
3425 <ccts:Definition>An area which is an administrative subdivision within a
3426 country, such as a state, a county, a canton or a
3427 province.</ccts:Definition>
3428 <ccts:ObjectClassTerm>Country Subdivision</ccts:ObjectClassTerm>
3429 <ccts:ObjectClassTermQualifier>Administrative</ccts:ObjectClassTermQualifier>
3430 </xsd:documentation>
3431 </xsd:annotation>
3432 <xsd:sequence>
3433 <xsd:element name="ID" type="udt:IDType ">
3434 <xsd:annotation>
3435 <xsd:documentation xml:lang="en">
3436 <ccts:UniqueID>UN01009363</ccts:UniqueID>
3437 <ccts:Acronym>BBIE</ccts:Acronym>
3438 <ccts:DictionaryEntryName>Administrative_ Country Subdivision.
3439 Identification. Identifier</ccts:DictionaryEntryName>
3440 <ccts:Version>1.0</ccts:Version>
3441 <ccts:Definition>The identifier for this administrative country
3442 subdivision.</ccts:Definition>
3443 <ccts:Cardinality>1</ccts:Cardinality>
3444 <ccts:ObjectClassTerm>Country Subdivision</ccts:ObjectClassTerm>
3445 <ccts:ObjectClassTermQualifier>Administrative</ccts:ObjectClassTermQualifi
3446 er>
3447 <ccts:PropertyTerm>Identification</ccts:PropertyTerm>
3448 <ccts:PrimaryRepresentationTerm>Identifier</ccts:PrimaryRepresen
3449 tationTerm>
3450 </xsd:documentation>
3451 </xsd:annotation>
3452 </xsd:element>
3453 <xsd:element name="Description" type="udt:TextType" minOccurs="0
3454 ">
3455 <xsd:annotation>

```

```

3456 <xsd:documentation xml:lang="en">
3457 <ccts:UniqueID>UN01009364</ccts:UniqueID>
3458 <ccts:Acronym>BBIE</ccts:Acronym>
3459 <ccts:DictionaryEntryName>Administrative_ Country Subdivision. Description.
3460 Text</ccts:DictionaryEntryName>
3461 <ccts:Version>1.0</ccts:Version>
3462 <ccts:Definition>The textual description for this
3463 administrative country subdivision.</ccts:Definition>
3464 <ccts:Cardinality>0..1</ccts:Cardinality>
3465 <ccts:ObjectClassTerm>Country Subdivision</ccts:ObjectClassTerm>
3466 <ccts:ObjectClassTermQualifier>Administrative</ccts:ObjectClassTermQualifier>
3467 <ccts:PropertyTerm>Description</ccts:PropertyTerm>
3468 <ccts:PrimaryRepresentationTerm>Text</ccts:PrimaryRepresentationTerm>
3469 </xsd:documentation>
3470 </xsd:annotation>
3471 </xsd:element>

```

## 3472 Example B-10: Complete Structure

```

3473 <?xml version="1.0" encoding="UTF-8"?>
3474 <!-- ===== -->
3475 <!-- ===== [SCHEMA MODULE TYPE] Schema Module ===== -->
3476 <!-- ===== -->
3477 <!--
3478 Schema agency: [SCHEMA AGENCY NAME]
3479 Schema version: [SCHEMA VERSION] Schema
3480 date: [DATE OF SCHEMA]
3481 [Code list name:] [NAME OF CODE LIST] [Code
3482 list agency:] [CODE LIST AGENCY] [Code
3483 list version:] [VERSION OF CODE LIST]
3484 [Identifier list name:] [NAME OF IDENTIFIER LIST]
3485 [Identifier list agency:] [IDENTIFIER LIST AGENCY]
3486 [Identifier list version:] [VERSION OF IDENTIFIER LIST]
3487
3488 ECE draws attention to the possibility that the practice or implementation of its
3489 outputs (which include but are not limited to Recommendations, norms, standards,
3490 guidelines and technical specifications) may involve the use of a claimed intellectual
3491 property right.
3492
3493 Each output is based on the contributions of participants in the UN/CEFACT process, who
3494 have agreed to waive enforcement of their intellectual property rights pursuant to the
3495 UN/CEFACT IPR Policy (document ECE/TRADE/C/CEFACT/2010/20/Rev.2 available at
3496 http://www.unece.org/cefact/cf_docs.html or from the ECE secretariat). ECE takes no
3497 position concerning the evidence, validity or applicability of any claimed intellectual
3498 property right or any other right that might be claimed by any third parties related to
3499 the implementation of its outputs. ECE makes no representation that it has made any
3500 investigation or effort to evaluate any such rights.
3501
3502 Implementers of UN/CEFACT outputs are cautioned that any third-party intellectual
3503 property rights claims related to their use of a UN/CEFACT output will be their
3504 responsibility and are urged to ensure that their use of UN/CEFACT outputs does not
3505 infringe on an intellectual property right of a third party.
3506
3507 ECE does not accept any liability for any possible infringement of a claimed
3508 intellectual property right or any other right that might be claimed to relate to the
3509 implementation of any of its outputs.-->
3510
3511 <xsd:schema
3512 targetNamespace="urn:un:unece:unefact:data:draft:[MODULENAME]:[VERSION]"
3513 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3514 ... FURTHER NAMESPACES ...
3515 elementFormDefault="qualified" attributeFormDefault="unqualified">
3516 <!-- ===== -->
3517 <!-- ===== Include ===== -->
3518 <!-- ===== -->
3519 <!-- ===== Inclusion of [TYPE OF MODULE] ===== -->
3520 <!-- ===== -->
3521 <xsd:include namespace="..." schemaLocation="..."/>
3522 <!-- ===== -->
3523 <!-- ===== Imports ===== -->
3524 <!-- ===== -->
3525 <!-- ===== Import of [TYPE OF MODULE] ===== -->
3526 <!-- ===== -->
3527 <xsd:import namespace="..." schemaLocation="..."/>
3528 <!-- ===== -->

```



3524  
3525  
3526  
3527  
3528  
3529  
3530  
3531  
3532  
3533  
3534  
3535  
3536  
3537  
  
3538  
3539  
3540  
3541  
3542  
  
3543

```
<!-- ===== Element Declarations ===== -->
<!-- =====
3526 <!-- ===== Root element ===== -->
3527 <!-- =====
3528 <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]">
3529 <!-- =====
3530 <!-- ===== Global Element Declarations ===== -->
3531 <!-- =====
3532 <xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]">
3533 <!-- =====
3534 <!-- ===== Type Definitions ===== -->
3535 <!-- =====
3536 <!-- ===== Type Definition: [TYPE] ===== -->
3537 <!-- =====

<xsd:complexType name="[TYPENAME]">
  <xsd:restriction base="xsd:token">
    ... see type definition ....
  </xsd:restriction>
</xsd:complexType>
```

3544

## Appendix C BPS Approved Acronyms and Abbreviations

3545

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

3546

3547

**ID – Identifier**

3548

**URI – Uniform Resource Identifier**

3549

## Appendix D Common Use Cases for Code Lists and Identifier Lists

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

### D.1 The Use of Code Lists within XML Schemas

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

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

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

#### Example D-1: Code Use Example Schema

```
<xsd:schema xmlns:ram="urn:un:unece:cefact:ram:0p1"
xmlns:udt="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:1"
xmlns:qdt="urn:un:unece:uncefact:data:draft:QualifiedDataTypeSchemaModule:1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:un:unece:cefact:ram:1p1"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- Imports -->
  <xsd:import
namespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:1"
schemaLocation=" http://www.unece.org/uncefact/data/draft/unqualifieddatatype_1.xsd"/>
  <xsd:import
namespace="urn:un:unece:uncefact:data:draft:QualifiedDataTypeSchemaModule:1"
schemaLocation=" http://www.unece.org/uncefact/data/draft/qualifieddatatype_1.xsd"/>
  <!-- Root element -->
  <xsd:element name="PurchaseOrderRequest" type="ram:PurchaseOrderRequestType"/>
  <!-- Messase type declaration -->
  <xsd:complexType name="PurchaseOrderRequestType">
    <xsd:sequence>
      <xsd:element name="Product" type="ram:ProductType"/>
    </xsd:sequence>
  </xsd:complexType>
  <!-- The below type declaration would normally appear in a separate schema module for all
reusable components (ABIE) but is included here for completeness -->
  <xsd:complexType name="ProductType">
    <xsd:sequence>
      <xsd:element name="TotalAmount" type="udt:AmountType"/>
      <xsd:element name="TaxCurrencyCode" type="udt:CodeType"/>
      <xsd:element name="ChangeCurrencyCode" type="qdt:CurrencyCodeType"/>
      <xsd:element name="CalculationCurrencyCode"
type="qdt:CalculationCurrencyCodeType"/>
      <xsd:element name="RestrictedCurrencyCode" type="qdt:RestrictedCurrencyCodeType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

This schema example imports:

- the schema module of all unqualified data types, such as, `udt:AmountType`, `udt:CodeType`, `udt:QuantityType`.
- the schema module of all qualified data types, in which the two specific data types `CurrencyCodeType` and `CalculationCurrencyCodeType` are defined.

3607 Within the **xsd:complexType** of **ProductType**, five local elements are declared. Each of these elements  
3608 represent one of the five different code list options.

## 3609 D.1.1 Referencing a Predefined Standard Code List in and Unqualified Data 3610 Type

3611 In the Code Use Example Schema, the element **TotalAmount** is declared as:

```
3612 <xsd:element name="TotalAmount" type="udt:AmountType"/>
```

3613 As shown in the element declaration, **TotalAmount** is of the CCTS unqualified data type **udt:AmountType**  
3614 which has been defined in the UN/CEFACT unqualified data type schema module (See Section 7.6). The  
3615 **udt:AmountType** declaration in the unqualified schema module is as follows:

```
3616 <xsd:schema  
3617 targetNamespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:1"  
3618 xmlns:clm54217="urn:un:unece:uncefact:codelist:draft:5:4217:2001" ...  
3619 elementFormDefault="qualified" attributeFormDefault="unqualified">  
3620 <!-- =====  
3621 <!-- ===== Imports  
3622 <!-- =====  
3623 <!-- ===== Imports of Code Lists  
3624 <!-- =====  
3625 <xsd:import namespace="urn:un:unece:uncefact:codelist:draft:5:4217:2001"  
3626 schemaLocation=" http://www.unece.org/uncefact/codelist/draft/5/4217_2001_.xsd "/>  
3627 <!-- =====  
3628 <!-- ===== Type Definitions  
3629 <!-- =====  
3630 <!-- ===== Amount. Type  
3631 <!-- =====  
3632 <xsd:complexType name="AmountType">  
3633 <xsd:simpleContent>  
3634 <xsd:extension base="xsd:decimal">  
3635 <xsd:attribute name="currencyID" type="clm54217:CurrencyCodeContentType"  
3636 use="required"/>  
3637 </xsd:extension>  
3638 </xsd:simpleContent>  
3639 </xsd:complexType>
```

3640 This **udt:AmountType** has attributes declared that represent the supplementary components defined in  
3641 CCTS for this data type. These attributes include **currencyCode** for the supplementary component of  
3642 **Amount. Currency. Code**. This **currencyCode** attribute is declared to be of the **xsd:simpleType**  
3643 **clm54217:CurrencyCodeContentType**. The **clm54217:CurrencyCodeContentType** has been  
3644 declared in the code list schema module for ISO Currency Codes, and the allowed code values for the  
3645 **currencyCode** attribute have been defined as enumeration facets in the  
3646 **clm54217:CurrencyCodeContentType** type definition.

3647 An extract of the code list schema module for ISO Currency Codes is as follows:

```
3648 <!-- =====  
3649 <!-- ===== Root Element Declarations  
3650 <!-- =====  
3651 <xsd:element name="CurrencyCode" type="clm54217:CurrencyCodeContentType"/>  
3652 <!-- =====  
3653 <!-- ===== Type Definitions  
3654 <!-- =====  
3655 <!-- ===== Code List Type Definition: Country Codes  
3656 <!-- =====  
3657 <xsd:simpleType name="CurrencyCodeContentType">  
3658 <xsd:restriction base="xsd:token">  
3659 <xsd:enumeration value="AED">  
3660 <xsd:annotation>  
3661 <xsd:documentation>  
3662 <CodeName>Dirham</CodeName>  
3663 </xsd:documentation>  
3664 </xsd:annotation>  
3665 </xsd:enumeration>  
3666 <xsd:enumeration value="AFN">  
3667 <xsd:annotation>  
3668 <xsd:documentation>
```

3669  
3670  
3671  
3672  
3673  
3674  
3675

```
<CodeName>Afghani</CodeName>
  </xsd:documentation>
</xsd:annotation>
</xsd:enumeration>
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

3676  
3677  
3678  
3679

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

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

3680  
3681

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

3682  
3683

## D.1.2 Referencing Any Code List Using the Unqualified Data Type **udt:CodeType**

3684  
3685

The second element in this example message – **TaxCurrencyCode** – is of the unqualified data type **udt:CodeType**.

3686

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

3687  
3688

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

3689

The **udt:CodeType** is declared in the unqualified schema module as:

3690  
3691  
3692  
3693  
3694  
3695  
3696  
3697  
3698  
3699  
3700  
3701

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

3702  
3703  
3704  
3705  
3706

When the **udt:CodeType** is used, either the **listURI** (which will point uniquely to the code list) should be used, or a combination of the other attributes should be used. Thus, it is possible to refer to the code list relevant attributes either by the specific attributes for the explicit display of supplementary components, or by the list URI in which the value is based on the namespace name conventions, such as:  
**urn:un:unece:uncefact:codelist:draft:5:4217:2001**.

3707  
3708

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

3709  
3710

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

3711

or

3712  
3713

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

3714  
3715

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

3716  
3717

## D.1.3 Referencing a Predefined Code List by Declaring a Specific Qualified Data Type

3718  
3719

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

3720

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

3721 The `qdt:CurrencyCodeType` would be defined in the qualified data type schema module as:

```
3722 <xsd:simpleType name="CurrencyCodeType">
3723   <xsd:restriction base="clm54217-A:CurrencyCodeContentType"/>
3724 </xsd:simpleType>
```

3725 This means that the value of the `ChangeCurrencyCode` element can only have code values from the  
3726 identified ISO 4217 code list. In an instance document this element would be represented as:

```
3727 <ChangeCurrencyCode>AED</ChangeCurrencyCode>
```

3728 It should be noted that when using this option no information about the code list to be used is carried in the  
3729 instance document as this is already defined in the XML schema.

## 3730 D.1.4 Choosing or Combining Values from Different Code Lists

3731 The fourth option is to chose or combine values from diverse code lists by using either the `xsd:choice` or  
3732 `xsd:union` elements.

### 3733 D.1.4.1 Choice

3734 In the Code Use Example Schema, the element `CalculationCurrencyCode` is declared as:

```
3735 <xsd:element name="CalculationCurrencyCode" type="qdt:CalculationCurrencyCodeType"/>
```

3736 The `CalculationCurrencyCode` element is of qualified data type  
3737 `qdt:CalculationCurrencyCodeType`.

3738 The `qdt:CalculationCurrencyCodeType` is defined in the qualified data type module as:

```
3739 <xsd:complexType name="CalculationCurrencyCodeType">
3740   <xsd:choice>
3741     <xsd:element ref="clm54217-N:CurrencyCode"/>
3742     <xsd:element ref="clm54217-A:CurrencyCode"/>
3743   </xsd:choice>
3744 </xsd:complexType>
```

3745 The `xsd:choice` element provides a choice of values from either the `clm54217-N:CurrencyCode` or from  
3746 `clm54217-A:CurrencyCode`. The schema module for `clm54217-A:CurrencyCode` is the same as the  
3747 one used in section 9.1.1 above. The sample schema module for `clm54217-N:CurrencyCode` is as  
3748 follows:

#### 3749 Example D-2: Sample `clm54217-N:CurrencyCode` Schema Module:

```
3750 <!-- ===== -->
3751 <!-- ===== Root Element Declarations ===== -->
3752 <!-- ===== -->
3753 <xsd:element name="CurrencyCode" type="clm54217-N:CurrencyCodeContentType"/>
3754 <!-- ===== Type Definitions ===== -->
3755 <!-- ===== -->
3756 <!-- ===== Code List Type Definition: 4217-N Currency Codes ===== -->
3757 <!-- ===== -->
3758 <xsd:simpleType name="CurrencyCodeContentType">
3759   <xsd:restriction base="xsd:token">
3760     <xsd:enumeration value="840">
3761       <xsd:annotation>
3762         <xsd:documentation>
3763           <CodeName>US Dollar</CodeName>
3764         </xsd:documentation>
3765       </xsd:annotation>
3766     </xsd:enumeration>
3767     <xsd:enumeration value="978">
3768       <xsd:annotation>
3769         <xsd:documentation>
3770           <CodeName>Euro</CodeName>
3771         </xsd:documentation>
3772       </xsd:annotation>
3773     </xsd:enumeration>
3774   </xsd:restriction>
3775 </xsd:simpleType>
3776 </xsd:schema>
```

3777 This `xsd:choice` option allows for the use of code values from different pre-defined code lists in the  
3778 instance document. The specific code list being used in the instance document will be represented by the  
3779 namespace prefix (`c1m54217-A` or `c1m54217-N`) being used for the namespace declaration of the imported  
3780 code list and for the `CurrencyCode` element:

```
3781 <PurchaseOrder ... xmlns:c1m54217-N="urn:un:unece:uncefact:codelist:draft:5:4217-N:2001"  
3782 ... >  
3783 <CalculationCurrencyCode>  
3784 <c1m54217-N:CurrencyCode>840</c1m54217-N:CurrencyCode>  
3785 </CalculationCurrencyCode>  
3786 ...  
3787 </PurchaseOrder>
```

3788 The namespace prefix unambiguously identifies to the recipient of the instance from which code list each  
3789 code value is defined.

#### 3790 **D.1.4.2 Union**

3791 The `xsd:union` code list approach is similar to that for the `xsd:choice` approach in that multiple code  
3792 lists are being used. The element declaration in the schema would be identical to that for choice in that the  
3793 element `CalculationCurrencyCode` is still based on the qualified data type  
3794 `qdt:CalculationCurrencyCodeType`.

```
3795 <xsd:element name="CalculationCurrencyCode" type="qdt:CalculationCurrencyCodeType"/>
```

3796 The difference is that the `qdt:calculationCurrencyCodeType` would be defined in the qualified data  
3797 type module using an `xsd:union` element rather than an `xsd:choice` element:

```
3798 <xsd:simpleType name="CalculationCurrencyCodeType">  
3799 <xsd:union memberTypes="c1m54217-N:CurrencyCodeContentType c1m54217-  
3800 A:CurrencyCodeContentType"/>  
3801 </xsd:simpleType>
```

3802 Here the declaration enables the instance to select a choice of values from either the `c1m54217-N:CurrencyCodeContentType` or from the `c1m54217-A:CurrencyCodeContentType`. The code list  
3803 schema module for `c1m54217-A:CurrencyCodeContentType` is the same as the one used in Section  
3804 D.1.1 above. The code list schema module for `c1m54217-N:CurrencyCodeContentType` is the same  
3805 as the one used in Section D.1.4.1.  
3806

3807 This `xsd:union` option allows for the use of code values from different pre-defined code lists in the  
3808 instance document. The code lists must be imported once in the XML schema module and must be shown  
3809 once in the XML instance. The specific code list will be represented by the namespace prefix (`c1m54217-A`  
3810 or `c1m54217-N`), but unlike the choice option, the element in the instance document will not have the  
3811 specific code list token conveyed as the first part of the element name. The recipient of the instance does  
3812 not know unambiguously in which code list each code value is defined. This is because a reference to the  
3813 specific code lists comes from different code list schema modules, such as, `c1m54217-N` and `c1m54217-A`.  
3814

3815 In an instance document this element could be represented as:

```
3816 <PurchaseOrder >  
3817 ...  
3818 <CalculationCurrencyCode>840</CalculationCurrencyCode>  
3819 ...  
3820 </PurchaseOrder>
```

3821 The advantage of the `xsd:union` approach is that attributes can make use of these code lists. For example,  
3822 it may make sense for an implementation to standardize across the board on two currency code lists and  
3823 have those apply to all of the data types, like `udt:AmountType` and its `currencyID` attribute.

#### 3824 **D.1.5 Restricting Allowed Code Values**

3825 This option is used when it is desired to reduce the number of allowed code values from an existing code list.  
3826 For example, a trading partner community may only recognize certain code values from the ISO 4217  
3827 Currency Code list. To accomplish this, three options exist:

- 3828 • Use `xsd:substitutionGroup` to replace the simple type that conveys the enumerated list of  
3829 codes



- 3830 • Use `xsd:redefine` to replace the simple type that conveys the enumerated list of codes
- 3831 • Create a new `xsd:simpleType` with the restricted set of value declarations

3832 The `xsd:substitutionGroup` and `xsd:redefine` features are specifically prohibited in the UN/CEFACT  
 3833 XML NDR due to issues associated with authentication, non-repudiation, ease of understanding, and tool  
 3834 support. Accordingly, when a user community wishes to restrict the allowed code values expressed in an  
 3835 existing schema, a new qualified datatype will be created in the QDT schema module, a new restricted  
 3836 codelist schema module will be created, and a new `xsd:simpleType` will be defined. This new  
 3837 `xsd:simpleType` will contain a complete list of allowed enumerations.

3838 In the example in section D.1.1, a `CurrencyID` element was declared and this element was of the  
 3839 `xsd:simpleType qdt:CurrencyCodeContentType` defined for currency code:

3840 If we wished to restrict the allowed values of `qdt:CurrencyCodeType`, we will have to define a new  
 3841 restricted datatype. For our example, this is the `qdt:RestrictedCurrencyCodeType`. Although in our  
 3842 data model this is a restriction of the `qdt:CurrencyCodeType`, this new datatype's restriction  
 3843 declaration will have a base value of `xsd:token` rather than `CurrencyCodeType` because of XSD  
 3844 limitations. In XSD, enumerations are repeating facets and the nature of `xsd:restriction` is such that  
 3845 the set of facets in a restricted type is the sum of the facets for the original type and the restricted type –  
 3846 actually resulting in an extension rather than restriction. For our example, the new `xsd:simpleType`  
 3847 definition would occur in a new code list schema module:

```

3848 <xsd:simpleType name="RestrictedCurrencyCodeContentType">
3849   <xsd:restriction base="xsd:token">
3850     <xsd:enumeration value="AED">
3851       <xsd:annotation>
3852         <xsd:documentation>
3853           <CodeName>Dirham</CodeName>
3854         </xsd:documentation>
3855       </xsd:annotation>
3856     </xsd:enumeration>
3857   </xsd:restriction>
3858 </xsd:simpleType>
  
```

3859 In the instance documents, allowed values of the element `RestrictedCurrencyCode` are limited to those  
 3860 contained in the restricted code list schema module.

## 3861 D.2 The Use of Identifier Schemes within XML Schemas

3862 The UN/CEFACT XML NDR allows for five alternative uses for identifier schemes:

- 3863 • Referencing a predefined standard identifier scheme, such as agency identifiers according to DE  
 3864 3055, as a supplementary component in an unqualified data type, such as `udt:codeType`.
- 3865 • Referencing any identifier scheme, standard or proprietary, by providing the required identification  
 3866 as attributes in the unqualified data type `udt:IdentifierType`
- 3867 • Referencing a predefined identifier scheme by declaring a specific qualified data type
- 3868 • Choosing or combining values from several identifier schemes
- 3869 • Restricting allowed identifier values

3870 The rules for identifier schemes are the same as those for code lists, thus the examples found in D.1 also  
 3871 apply to identifier lists

3872

3873

## Appendix E Annotation Templates

3874 The following templates define the annotation for each of the schema modules.

```
3875 <!-- Root Schema Documentation -->
3876 <xsd:annotation>
3877 <xsd:documentation xml:lang="en">
3878 <ccts:UniqueID></ccts:UniqueID>
3879 <ccts:Acronym>RSM</ccts:Acronym>
3880 <ccts:Name></ccts:Name>
3881 <ccts:Version></ccts:Version>
3882 <ccts:Definition></ccts:Definition>
3883 <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3884 <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3885 <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
3886 <ccts:ProductContextValue></ccts:ProductContextValue>
3887 <ccts:IndustryContextValue></ccts:IndustryContextValue>
3888 <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3889 <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3890 <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3891 </xsd:documentation>
3892 </xsd:annotation>

3893 <!-- ABIE Documentation -->
3894 <xsd:annotation>
3895 <xsd:documentation xml:lang="en">
3896 <ccts:UniqueID></ccts:UniqueID>
3897 <ccts:Acronym>ABIE</ccts:Acronym>
3898 <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3899 <ccts:Version></ccts:Version>
3900 <ccts:Definition></ccts:Definition>
3901 <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
3902 <ccts:ObjectClassQualifierTerm></ccts:ObjectClassQualifierTerm>
3903 <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3904 <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3905 <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextVale>
3906 <ccts:ProductContextValue></ccts:ProductContextValue>
3907 <ccts:IndustryContextValue></ccts:IndustryContextValue>
3908 <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3909 <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3910 <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3911 <ccts:UsageRule></ccts:UsageRule>
3912 <ccts:BusinessTerm></ccts:BusinessTerm>
3913 <ccts:Example></ccts:Example>
3914 </xsd:documentation>
3915 </xsd:annotation>

3916 <!-- BBIE Documentation -->
3917 <xsd:annotation>
3918 <xsd:documentation xml:lang="en">
3919 <ccts:UniqueID></ccts:UniqueID>
3920 <ccts:Acronym>ABIE</ccts:Acronym>
3921 <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3922 <ccts:Version></ccts:Version>
3923 <ccts:Definition></ccts:Definition>
3924 <ccts:Cardinality></ccts: Cardinality>
3925 <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
3926 <ccts:ObjectClassQualifierTerm></ccts:ObjectClassQualifierTerm>
3927 <ccts:PropertyTerm></ccts:PropertyTerm>
3928 <ccts:PropertyQualifierTerm></ccts:PropertyQualifierTerm>
```

```

3929     <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm
3930     <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3931     <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3932     <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
3933     <ccts:ProductContextValue></ccts:ProductContextValue>
3934     <ccts:IndustryContextValue></ccts:IndustryContextValue>
3935     <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3936     <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3937     <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3938     <ccts:UsageRule></ccts:UsageRule>
3939     <ccts:BusinessTerm></ccts:BusinessTerm>
3940     <ccts:Example></ccts:Example>
3941   </xsd:documentation>
3942 </xsd:annotation>

3943 <!-- ASBIE Documentation -->
3944 <xsd:annotation>
3945   <xsd:documentation xml:lang="en">
3946     <ccts:UniqueID></ccts:UniqueID>
3947     <ccts:Acronym>ABIE</ccts:Acronym>
3948     <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3949     <ccts:Version></ccts:Version>
3950     <ccts:Definition></ccts:Definition>
3951     <ccts:Cardinality></ccts:Cardinality>
3952     <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
3953     <ccts:ObjectClassQualifierTerm></ccts:ObjectClassQualifierTerm>
3954     <ccts:PropertyTerm></ccts:PropertyTerm>
3955     <ccts:PropertyQualifierTerm></ccts:PropertyQualifierTerm>
3956     <ccts:AssociatedObjectClassTerm></ccts:AssociatedObjectClassTerm>
3957     <ccts:AssociatedObjectClassQualifierTerm></ccts:AssociatedObjectClassQualifierTerm>
3958     <ccts:AssociationType></ccts:AssociationType>
3959     <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3960     <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3961     <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
3962     <ccts:ProductContextValue></ccts:ProductContextValue>
3963     <ccts:IndustryContextValue></ccts:IndustryContextValue>
3964     <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3965     <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3966     <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3967     <ccts:UsageRule></ccts:UsageRule>
3968     <ccts:BusinessTerm></ccts:BusinessTerm>
3969     <ccts:Example></ccts:Example>
3970   </xsd:documentation>
3971 </xsd:annotation>

3972 <!-- Qualified Data Type Documentation -->
3973 <xsd:annotation>
3974   <xsd:documentation xml:lang="en">
3975     <ccts:UniqueID></ccts:UniqueID>
3976     <ccts:Acronym>QDT</ccts:Acronym>
3977     <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3978     <ccts:Version></ccts:Version>
3979     <ccts:Definition></ccts:Definition>
3980     <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
3981     <ccts:DataTypeQualifierTerm></ccts:DataTypeQualifierTerm>
3982     <ccts:PrimitiveType></ccts:PrimitiveType>
3983     <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3984     <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3985     <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>

```

```

3986         <ccts:ProductContextValue></ccts:ProductContextValue>
3987         <ccts:IndustryContextValue></ccts:IndustryContextValue>
3988         <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3989         <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3990         <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3991         <ccts:UsageRule></ccts:UsageRule>
3992         <ccts:BusinessTerm></ccts:BusinessTerm>
3993         <ccts:Example></ccts:Example>
3994     </xsd:documentation>
3995 </xsd:annotation>

3996 <!-- Qualified Data Type Supplementary Component Documentation-->
3997 <xsd:annotation>
3998     <xsd:documentation xml:lang="en">
3999         <ccts:UniqueID></ccts:UniqueID>
4000         <ccts:Acronym>SC</ccts:Acronym>
4001         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4002         <ccts:Definition></ccts:Definition>
4003         <ccts:Cardinality></ccts: Cardinality>
4004         <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
4005         <ccts:PropertyTerm></ccts:PropertyTerm>
4006         <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4007         <ccts:PrimitiveType></ccts:PrimitiveType>
4008         <ccts:UsageRule></ccts:UsageRule>
4009     </xsd:documentation>
4010 </xsd:annotation>

4011 <!-- Unqualified Data Type Documentation-->
4012 <xsd:annotation>
4013     <xsd:documentation xml:lang="en">
4014         <ccts:UniqueID></ccts:UniqueID>
4015         <ccts:Acronym>UDT</ccts:Acronym>
4016         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4017         <ccts:Version></ccts:Version>
4018         <ccts:Definition></ccts:Definition>
4019         <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4020         <ccts:PrimitiveType></ccts:PrimitiveType>
4021         <ccts:UsageRule></ccts:UsageRule>
4022     </xsd:documentation>
4023 </xsd:annotation>

4024 <!-- Unqualified Data Type Supplementary Component Documentation-->
4025 <xsd:annotation>
4026     <xsd:documentation xml:lang="en">
4027         <ccts:UniqueID></ccts:UniqueID>
4028         <ccts:Acronym>SC</ccts:Acronym>
4029         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4030         <ccts:Definition></ccts:Definition>
4031         <ccts:Cardinality></ccts: Cardinality>
4032         <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
4033         <ccts:PropertyTerm></ccts:PropertyTerm>
4034         <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4035         <ccts:PrimitiveType></ccts:PrimitiveType>
4036         <ccts:UsageRule></ccts:UsageRule>
4037     </xsd:documentation>
4038 </xsd:annotation>

4039 <!-- Core Component Type Documentation -->
4040 <xsd:annotation>
4041     <xsd:documentation xml:lang="en">

```

```

4042         <ccts:UniqueID></ccts:UniqueID>
4043         <ccts:Acronym>CCT</ccts:Acronym>
4044         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4045         <ccts:Version></ccts:Version>
4046         <ccts:Definition></ccts:Definition>
4047         <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4048         <ccts:PrimitiveType></ccts:PrimitiveType>
4049         <ccts:UsageRule></ccts:UsageRule>
4050     </xsd:documentation>
4051 </xsd:annotation>
4052 <!-- Core Component Type Supplementary Component Documentation-->
4053 <xsd:annotation>
4054     <xsd:documentation xml:lang="en">
4055         <ccts:UniqueID></ccts:UniqueID>
4056         <ccts:Acronym>SC</ccts:Acronym>
4057         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4058         <ccts:Definition></ccts:Definition>
4059         <ccts:Cardinality></ccts:Cardinality>
4060         <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
4061         <ccts:PropertyTerm></ccts:PropertyTerm>
4062         <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4063         <ccts:PrimitiveType></ccts:PrimitiveType>
4064         <ccts:UsageRule></ccts:UsageRule>
4065     </xsd:documentation>
4066 </xsd:annotation>
4067 <!-- Code List / Identification Schema Documentation-->
4068 <xsd:annotation>
4069     <xsd:documentation xml:lang="en">
4070         <ccts:Name></ccts:Name>
4071         <ccts:Description></ccts:Description>
4072     </xsd:documentation>
4073 </xsd:annotation>
4074

```

## Appendix F Naming and Design Rules Checklist

4075

- 4076 [R1] Conformance shall be determined through adherence to the content of normative sections,  
4077 rules and definitions.
- 4078 [R2] All UN/CEFACT XSD Schema design rules MUST be based on the *W3C XML*  
4079 *SchemaRecommendations: XML Schema Part 1: Structures and XML Schema Part 2: Data Types*
- 4080 [R3] All UN/CEFACT XSD Schema and UN/CEFACT conformant XML instance documents MUST be  
4081 based on the W3C suite of technical specifications holding recommendation status.
- 4082 [R4] UN/CEFACT XSD Schema MUST follow the standard structure defined in Appendix B
- 4083 [R5] Each element or attribute XML name MUST have one and only one fully qualified  
4084 XPath(FQXP)
- 4085 [R6] Element, attribute and type names MUST be composed of words in the English language,  
4086 using the primary English spellings provided in the Oxford English Dictionary.
- 4087 [R7] Lower camel case (LCC) MUST be used for naming attributes
- 4088 [R8] Upper camel case (UCC) MUST be used for naming elements and types.
- 4089 [R9] Element, attribute and type names MUST be in singular form unless the concept itself is  
4090 plural.
- 4091 [R10] Element, attribute and type names MUST be drawn from the following character set: **a-z**  
4092 and **A-Z**. Any special characters such as spaces, underscores, and periods that exist in  
4093 the underlying Dictionary Entry Names MUST be removed.
- 4094 [R11] This rule has been combined with [R10].
- 4095 [R12] XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other word  
4096 truncations, except those included in the UN/CEFACT controlled vocabulary or listed in Appendix  
4097 C.
- 4098 [R13] The acronyms and abbreviations listed in Appendix C MUST always be used.
- 4099 [R14] Acronyms and abbreviations at the beginning of an attribute declaration MUST appear in all  
4100 lower case. All other acronym and abbreviation usage in an attribute declaration must appear in  
4101 upper case.
- 4102 [R15] Acronyms MUST appear in all upper case for all element declarations and type definitions.
- 4103 [R16] The schema module file name for modules other than code lists or identifier lists MUST be of the  
4104 form **<SchemaModuleName>\_<Version>.xsd**, with periods, spaces, or other separators and  
4105 the words Schema Module removed.
- 4106 [R17] The schema module file name for code lists and identifier lists, MUST be of the form  
4107 **<AgencyName>\_<ListName>\_<Version>.xsd**, with periods, spaces, or other separators  
4108 removed.
- 4109 [R18] In representing versioning schemes in file names, only the major version should be  
4110 included.
- 4111 [R19] A root schema MUST be created for each unique business information payload.
- 4112 [R20] Each UN/CEFACT root schema module MUST be named **<BusinessInformationPayload>**  
4113 **Schema Module**.
- 4114 [R21] A root schema MUST NOT replicate reusable constructs available in schema modules  
4115 capable of being referenced through **xsd:include** or **xsd:import**.
- 4116 [R22] UN/CEFACT XSD schema modules MUST either be treated as external schema modules, or as  
4117 internal schema modules of the root schema.
- 4118 [R23] All UN/CEFACT internal schema modules MUST be in the same namespace as their  
4119 corresponding **rsm:RootSchema**.
- 4120 [R24] Each UN/CEFACT internal schema module MUST be named  
4121 **<ParentRootSchemaModuleName><InternalSchemaModuleFunction> Schema Module**
- 4122 [R25] A Core Component Type schema module MUST be created.
- 4123 [R26] The **cct:CoreComponentType** schema module MUST be named 'Core Component Type  
4124 Schema Module'.
- 4125 [R203] An Unqualified Data Type MUST NOT contain any restriction on their source CCTs other than  
4126 those defined in CCTS and agreed upon best practices.
- 4127 [R27] An Unqualified Data Type schema module MUST be created
- 4128 [R28] The **udt:UnqualifiedDataType** schema module MUST be named 'Unqualified  
4129 Data TypeSchema Module'



- 4130 [R29] A Qualified Data Type schema module MUST be created.
- 4131 [R30] The `qdt:QualifiedDataType` schema module MUST be named 'Qualified Data Type Schema  
4132 Module'.
- 4133 [R31] A Reusable Aggregate Business Information Entity schema module MUST be created.
- 4134 [R32] The `ram:ReusableAggregateBusinessInformationEntity` schema module  
4135 MUST be named 'Reusable Aggregate Business Information Entity Schema Module'.
- 4136 [R33] Reusable Code List schema modules MUST be created to convey code list enumerations
- 4137 [R34] The name of each `clm:CodeList` schema module MUST be of the form: `<Code List  
4138 Agency Identifier|Code List Agency Name><Code List Identification  
4139 Identifier|Code List Name> - Code List Schema Module`  
4140 Where:  
4141 Code List Agency Identifier = Identifies the agency that maintains the code list  
4142 Code List Agency Name = Agency that maintains the code list  
4143 Code List Identification Identifier = Identifies a list of the respective corresponding codes  
4144 Code List Name = The name of the code list as assigned by the agency that maintains the  
4145 code list
- 4146 [R35] An identifier list schema module MUST be created to convey enumerated values for each  
4147 identifier list that requires runtime validation.
- 4148 [R36] The name of each `ids:IdentifierList` schema module MUST be of the form:  
4149 `<Identifier Scheme Agency Identifier|Identifier Scheme Agency  
4150 Name><Identifier Scheme Identifier|Identifier Scheme Name> -  
4151 Identifier List Schema Module`  
4152 Where:  
4153 Identifier Scheme Agency Identifier = The identification of the agency that maintains the  
4154 identifier list  
4155 Identifier Scheme Agency Name = Agency that maintains the identifier list  
4156 Identifier Scheme Identifier = The identification of the identifier list  
4157 Identification Scheme Name = Name as assigned by the agency that maintains the identifier list
- 4158 [R37] Imported schema modules MUST be fully conformant with the UN/CEFACT *XML Naming and  
4159 Design Rules* Technical Specification and the UN/CEFACT *Core Components Technical  
4160 Specification*.
- 4161 [R38] Every UN/CEFACT defined or imported schema module MUST have a namespace declared,  
4162 using the `xsd:targetNamespace` attribute.
- 4163 [R39] Every version of a defined or imported schema module other than internal schema modules  
4164 MUST have its own unique namespace.
- 4165 [R40] UN/CEFACT published namespace declarations MUST NOT be changed, and its contents MUST  
4166 NOT be changed unless such change does not break backward compatibility.
- 4167 [R41] UN/CEFACT namespaces MUST be defined as Uniform Resource Names.
- 4168 [R42] The names for namespaces MUST have the following structure while the schema is at draft  
4169 status:  
4170 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>`  
4171 Where:  
4172 `schematype` = a token identifying the type of schema module:  
4173 `data|process|codelist|identifierlist|documentation`  
4174 `status` = a token identifying the standards status of the schema module: `draft|standard`  
4175 `name` = the name of the schema module (using upper camel case) with periods, spaces, or  
4176 other separators and the words 'schema module' removed.  
4177 `major` = the major version number. Sequentially assigned, first release starting with the number 1.
- 4178 [R43] This rule was combined with [R42].
- 4179 [R44] UN/CEFACT namespace values will only be assigned to UN/CEFACT developed objects.
- 4180 [R45] The general structure for schema location MUST be:  
4181 `../<schematype>/<status>/<name>_<major>.<minor>[p <revision>].xsd`  
4182 Where:  
4183 `schematype` = a token identifying the type of schema module:  
4184 `data|process|codelist|identifierlist|documentation`  
4185 `status` = the status of the schema as: `draft|standard`



- 4186 name = the name of the schema module (using upper camel case) with periods, spaces, or  
 4187 other separators and the words 'schema module' removed.  
 4188 major = the major version number, sequentially assigned, first release starting with the number 1.  
 4189 minor = the minor version number within a major release, sequentially assigned, first release  
 4190 starting with the number 0.  
 4191 revision = sequentially assigned alphanumeric character for each revision of a minor release. Only  
 4192 applicable where status = draft.
- [R46] Each `xsd:schemaLocation` attribute declaration MUST contain a resolvable URL, and in the  
 4193 case of an absolute path, a persistent URL.
- [R47] This rule has been removed.
- [R48] The `xsd:schema` version attribute MUST always be declared.
- [R49] The `xsd:schema` version attribute MUST use the following template:  
 4198 `<xsd:schema ... version="<major>.<minor>">`
- [R50] Every schema version namespace declaration MUST have the URI of:  
 4200 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>`
- [R51] Every UN/CEFACT XSD Schema and schema module major version number MUST be a  
 4202 sequentially assigned incremental integer greater than zero.
- [R52] Minor versioning MUST be limited to declaring new optional XSD constructs, extending  
 4203 existing XSD constructs, or refinements of an optional nature.
- [R53] For UN/CEFACT minor version changes, the name of the schema construct MUST NOT change.
- [R54] Changes in minor versions MUST NOT break semantic compatibility with prior versions  
 4206 having the same major version number.
- [R55] UN/CEFACT minor version schema MUST incorporate all XML constructs from the  
 4207 immediately preceding major or minor version schema.
- [R56] The `xsd:elementFormDefault` attribute MUST be declared and its value set to `qualified`.
- [R57] The `xsd:attributeFormDefault` attribute MUST be declared and its value set to  
 4212 `unqualified`.
- [R58] The `xsd` prefix MUST be used in all cases when referring to <http://www.w3.org/2001/XMLSchema> as follows:  
 4214 `xmlns:xsd=http://www.w3.org/2001/XMLSchema`.
- [R59] `xsd:appInfo` MUST NOT be used.
- [R60] `xsd:notation` MUST NOT be used.
- [R61] `xsd:wildcard` MUST NOT be used.
- [R62] The `xsd:any` element MUST NOT be used.
- [R63] The `xsd:any` attribute MUST NOT be used.
- [R64] Mixed content MUST NOT be used (excluding documentation).
- [R65] `xsd:substitutionGroup` MUST NOT be used.
- [R66] `xsd:ID/xsd:IDREF` MUST NOT be used.
- [R67] `xsd:key/xsd:keyref` MUST be used for information association.
- [R68] The absence of a construct or data MUST NOT carry meaning.
- [R69] User declared attributes MUST only be used to convey core component type (CCT)  
 4226 supplementary component information.
- [R70] A `xsd:attribute` that represents a supplementary component with variable information MUST  
 4228 be based on the appropriate XSD built-in data type.
- [R71] A `xsd:attribute` that represents a supplementary component which represents codes MUST  
 4230 be based on the `xsd:simpleType` of the appropriate code list.
- [R72] A `xsd:attribute` that represents a supplementary component which represents identifiers  
 4232 MUST be based on the `xsd:simpleType` of the appropriate identifier scheme.
- [R73] The `xsd:nilable` attribute MUST NOT be used.
- [R74] Empty elements MUST NOT be used.
- [R75] Every BBIE leaf element declaration MUST be of the `udt:UnqualifiedDataType` or  
 4236 `qdt:QualifiedDataType` that represents the source basic business information entity  
 4237 (BBIE) data type.
- [R76] The `xsd:all` element MUST NOT be used.
- [R77] All type definitions MUST be named.

- 4241 [R78] Data type definitions with the same semantic meaning MUST NOT have an identical set of  
4242 facet restrictions.
- 4243 [R79] **xsd:extension** MUST only be used in the **cct:CoreComponentType** schema module and  
4244 the **udt:UnqualifiedDataType** schema module. When used it MUST only be used for  
4245 declaring **xsd:attributes** to accommodate relevant supplementary components.
- 4246 [R80] When **xsd:restriction** is applied to a **xsd:simpleType** or **xsd:complexType** that  
4247 represents a data type the derived construct MUST use a different name.
- 4248 [R81] Each UN/CEFACT defined or declared construct MUST use the **xsd:annotation** element for  
4249 required CCTS documentation.
- 4250 [R82] The root schema module MUST be represented by a unique token.
- 4251 [R83] The **rsm:RootSchema** MUST import the following schema modules:  
4252 – **ram:ReusableABIE** Schema Module  
4253 – **udt:UnqualifiedDataType** Schema Module  
4254 – **qdt:QualifiedDataType** Schema Module
- 4255 [R84] A **rsm:RootSchema** in one UN/CEFACT namespace that is dependent upon type definitions or  
4256 element declaration defined in another namespace MUST import the **rsm:RootSchema** from  
4257 that namespace.
- 4258 [R85] A **rsm:RootSchema** in one UN/CEFACT namespace that is dependent upon type definitions or  
4259 element declarations defined in another namespace MUST NOT import Schema Modules from  
4260 that namespace other than the **rsm:RootSchema**.
- 4261 [R86] The **rsm:RootSchema** MUST include any internal schema modules that reside in the root  
4262 schema namespace.
- 4263 [R87] A single global element known as the root element, representing the business information  
4264 payload, MUST be declared in a **rsm:RootSchema**.
- 4265 [R88] The name of the root element MUST be the name of the business information payload with  
4266 separators and spaces removed.
- 4267 [R89] The root element declaration must be of **xsd:complexType** that represents the business  
4268 information payload.
- 4269 [R90] Root schema MUST define a single **xsd:complexType** that fully describes the business  
4270 information payload.
- 4271 [R91] The name of the root schema **xsd:complexType** MUST be the name of the root element with  
4272 the word 'Type' appended.
- 4273 [R92] The **rsm:RootSchema** root element declaration MAY have a structured set of annotations  
4274 present in the following pattern:
- 4275 ○ UniqueID (required): The identifier that references the business information payload  
4276 instance in a unique and unambiguous way.
  - 4277 ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
4278 always be RSM.
  - 4279 ○ Name (required): The name of the business information payload.
  - 4280 ○ Version (required): An indication of the evolution over time of a business information  
4281 payload.
  - 4282 ○ Definition (required): A brief description of the business information payload.
  - 4283 ○ BusinessProcessContextValue (required, repetitive): The business process with which this  
4284 business information is associated.
  - 4285 ○ GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for  
4286 this business information payload.
  - 4287 ○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this  
4288 business information payload.
  - 4289 ○ ProductContextValue (optional, repetitive): The product context for this business information  
4290 payload.
  - 4291 ○ IndustryContextValue (optional, repetitive): The industry context for this business information  
4292 payload.
  - 4293 ○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this business  
4294 information payload.
  - 4295 ○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this  
4296 business information payload.

- 4297           ○ SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for  
4298           this business information payload.
- 4299 [R93] All UN/CEFACT internal schema modules MUST be in the same namespace as their  
4300           corresponding **rsm:RootSchema**.
- 4301 [R94] The internal schema module MUST be represented by the same token as its **rsm:RootSchema**.
- 4302 [R95] The Reusable Aggregate Business Information Entity schema module MUST be represented by  
4303           the token **ram**.
- 4304 [R96] The **ram:ReusableAggregateBusinessInformationEntity** schema MUST import the  
4305           following schema modules:  
4306           – **udt:UnqualifiedDataType** Schema Module  
4307           – **qdt:QualifiedDataType** Schema Module
- 4308 [R97] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named  
4309           **xsd:complexType** MUST be defined.
- 4310 [R98] The name of the ABIE **xsd:complexType** MUST be the **ccts:DictionaryEntryName** with  
4311           the spaces and separators removed, approved abbreviations and acronyms applied, and with the  
4312           'Details' suffix replaced with 'Type'.
- 4313 [R99] Every aggregate business information entity (ABIE) **xsd:complexType** definition content model  
4314           MUST use the **xsd:sequence** and/or **xsd:choice** elements to reflect each property (BBIE or  
4315           ASBIE) of its class.
- 4316 [R100] Recursion of **xsd:sequence** and/or **xsd:choice** MUST NOT occur.
- 4317 [R101] The order and cardinality of the elements within an ABIE **xsd:complexType** MUST be  
4318           according to the structure of the ABIE as defined in the model.
- 4319 [R102] For each ABIE, a named **xsd:element** MUST be globally declared.
- 4320 [R103] The name of the ABIE **xsd:element** MUST be the **ccts:DictionaryEntryName** with the  
4321           separators and 'Details' suffix removed and approved abbreviations and acronyms applied.
- 4322 [R104] Every ABIE global element declaration MUST be of the **xsd:complexType** that represents the  
4323           ABIE.
- 4324 [R105] For every BBIE identified in an ABIE, a named **xsd:element** MUST be locally declared within  
4325           the **xsd:complexType** representing that ABIE.
- 4326 [R106] Each BBIE element name declaration MUST be the property term and qualifiers and the  
4327           representation term of the basic business information entity (BBIE). Where the word  
4328           'identification' is the final word of the property term and the representation term is 'identifier',  
4329           the term 'identification' MUST be removed. Where the word 'indication' is the final word of  
4330           the property term and the representation term is 'indicator', the term 'indication' MUST be  
4331           removed from the property term.
- 4332 [R107] If the representation term of a BBIE is 'text', 'text' MUST be removed.
- 4333 [R108] The BBIE element MUST be based on an appropriate data type that is defined in the  
4334           UN/CEFACT **qdt:QualifiedDataType** or **udt:UnqualifiedDataType** schema  
4335           modules.
- 4336 [R109] For every ASBIE whose **ccts:AssociationType** is a composition, a named **xsd:element**  
4337           MUST be locally declared.
- 4338 [R110] For each locally declared ASBIE, the element name MUST be the ASBIE property term and  
4339           qualifier term(s) and the object class term and qualifier term(s) of the associated ABIE.
- 4340 [R111] For each locally declared ASBIE, the element declaration MUST be of the **sd:complexType** that  
4341           represents its associated ABIE.
- 4342 [R112] For every ASBIE whose **ccts:AssociationType** is not a composition, the globally  
4343           declared element for the associated ABIE must be referenced using **xsd:ref**.
- 4344 [R113] For every ABIE **xsd:complexType** and **xsd:element** definition a structured set of  
4345           annotations MAY be present in the following pattern:  
4346           ○ UniqueID (required): The identifier that references an ABIE instance in a unique and  
4347           unambiguous way.  
4348           ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
4349           always be ABIE.  
4350           ○ DictionaryEntryName (required): The official name of an ABIE.  
4351           ○ Version (required): An indication of the evolution over time of an ABIE instance.

- 4352 ○ Definition (required): The semantic meaning of an ABIE.
- 4353 ○ ObjectClassTerm (required): The Object Class Term of the ABIE.
- 4354 ○ ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the ABIE.
- 4355 ○ BusinessProcessContextValue (optional, repetitive): The business process with which this
- 4356 ABIE is associated.
- 4357 ○ GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for
- 4358 this ABIE.
- 4359 ○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
- 4360 ABIE.
- 4361 ○ ProductContextValue (optional, repetitive): The product context for this ABIE.
- 4362 ○ IndustryContextValue (optional, repetitive): The industry context for this ABIE.
- 4363 ○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this ABIE.
- 4364 ○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this ABIE.
- 4365 ○ SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
- 4366 this ABIE.
- 4367 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4368 applicable to the ABIE.
- 4369 ○ BusinessTerm (optional, repetitive): A synonym term under which the ABIE is commonly
- 4370 known and used in the business.
- 4371 ○ Example (optional, repetitive): Example of a possible value of an ABIE.
- 4372 [R114] This rule was combined with [R113].
- 4373 [R115] For every BBIE `xsd:element` declaration a structured set of annotations MAY be present in
- 4374 the following pattern:
- 4375 ○ UniqueID (required): The identifier that references a BBIE instance in a unique and
- 4376 unambiguous way.
- 4377 ○ Acronym (required): The abbreviation of the type of component. In this case the value will
- 4378 always be BBIE.
- 4379 ○ DictionaryEntryName (required): The official name of the BBIE.
- 4380 ○ VersionID (required): An indication of the evolution over time of a BBIE instance.
- 4381 ○ Definition (required): The semantic meaning of the BBIE.
- 4382 ○ Cardinality (required): Indication whether the BIE Property represents a not-applicable,
- 4383 optional, required and/or repetitive characteristic of the ABIE.
- 4384 ○ ObjectClassTerm (required): The Object Class Term of the parent ABIE.
- 4385 ○ ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the parent ABIE.
- 4386 ○ PropertyTerm (required): The Property Term of the BBIE.
- 4387 ○ PropertyQualifierTerm (optional): Qualifies the Property Term of the BBIE.
- 4388 ○ PrimaryRepresentationTerm (required): The Primary Representation Term of the BBIE.
- 4389 ○ BusinessProcessContextValue (optional, repetitive): The business process with which this
- 4390 BBIE is associated.
- 4391 ○ GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for
- 4392 this BBIE.
- 4393 ○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
- 4394 BBIE.
- 4395 ○ ProductContextValue (optional, repetitive): The product context for this BBIE.
- 4396 ○ IndustryContextValue (optional, repetitive): The industry context for this BBIE.
- 4397 ○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this BBIE.
- 4398 ○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this BBIE.
- 4399 ○ SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
- 4400 this BBIE.
- 4401 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4402 applicable to this BBIE.

- 4403           ○ BusinessTerm (optional, repetitive): A synonym term under which the BBIE is commonly  
4404           known and used in the business.
- 4405           ○ Example (optional, repetitive): Example of a possible value of a BBIE.
- 4406   [R116] For every ASBIE `xsd:element` declaration a structured set of annotations MAY be present in  
4407   the following pattern:
- 4408           ○ UniqueID (required): The identifier that references an ASBIE instance in a unique and  
4409           unambiguous way.
- 4410           ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
4411           always be ASBIE.
- 4412           ○ DictionaryEntryName (required): The official name of the ASBIE.
- 4413           ○ Version (required): An indication of the evolution over time of the ASBIE instance.
- 4414           ○ Definition (required): The semantic meaning of the ASBIE.
- 4415           ○ Cardinality (required): Indication whether the ASBIE Property represents a not-applicable,  
4416           optional, required and/or repetitive characteristic of the ABIE.
- 4417           ○ ObjectClassTerm (required): The Object Class Term of the associating ABIE.
- 4418           ○ ObjectClassQualifierTerm (optional): A term that qualifies the Object Class Term of the  
4419           associating ABIE.
- 4420           ○ PropertyTerm (required): The Property Term of the ASBIE.
- 4421           ○ PropertyQualifierTerm (Optional): A term that qualifies the Property Term of the ASBIE.
- 4422           ○ AssociatedObjectClassTerm (required): The Object Class Term of the associated ABIE.
- 4423           ○ AssociatedObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the  
4424           associated ABIE.
- 4425           ○ AssociationType (required): The Association Type of the ASBIE.
- 4426           ○ BusinessProcessContextValue (optional, repetitive): The business process with which this  
4427           ASBIE is associated.
- 4428           ○ GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for  
4429           this ASBIE.
- 4430           ○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this  
4431           ASBIE.
- 4432           ○ ProductContextValue (optional, repetitive): The product context for this ASBIE.
- 4433           ○ IndustryContextValue (optional, repetitive): The industry context for this ASBIE.
- 4434           ○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this ASBIE.
- 4435           ○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this  
4436           ASBIE.
- 4437           ○ SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for  
4438           this ASBIE.
- 4439           ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are  
4440           applicable to the ASBIE.
- 4441           ○ BusinessTerm (optional, repetitive): A synonym term under which the ASBIE is commonly  
4442           known and used in the business.
- 4443           ○ Example (optional, repetitive): Example of a possible value of an ASBIE.
- 4444   [R117] The core component type (CCT) schema module MUST be represented by the token `cct`.
- 4445   [R118] The `cct:CoreCoreComponentType` schema module MUST NOT include or import any  
4446   other schema modules.
- 4447   [R119] Every core component type MUST be defined as a named `xsd:complexType` in the  
4448   `cct:CoreComponentType` schema module.
- 4449   [R120] The name of each `xsd:complexType` based on a core component type MUST be the  
4450   dictionary entry name of the core component type (CCT), with the separators and spaces  
4451   removed and approved abbreviations applied.



- 4452 [R121] Each core component type `xsd:complexType` definition MUST contain one  
4453 `xsd:simpleContent` element.
- 4454 [R122] The core component type `xsd:complexType` definition `xsd:simpleContent` element MUST  
4455 contain one `xsd:extension` element. This `xsd:extension` element must include an XSD  
4456 based attribute that defines the specific XSD built-in data type required for the CCT content  
4457 component.
- 4458 [R123] Within the core component type `xsd:extension` element a `xsd:attribute` MUST be  
4459 declared for each supplementary component pertaining to that core component type.
- 4460 [R124] Each core component type supplementary component `xsd:attribute` name MUST be the  
4461 CCTS supplementary component dictionary entry name with the separators and spaces  
4462 removed.
- 4463 [R125] If the object class of the supplementary component dictionary entry name contains the name of  
4464 the representation term of the parent CCT, the duplicated object class word or words MUST be  
4465 removed from the supplementary component `xsd:attribute` name.
- 4466 [R126] If the object class of the supplementary component dictionary entry name contains the term  
4467 'identification', the term 'identification' MUST be removed from the supplementary component  
4468 `xsd:attribute` name.
- 4469 [R127] If the representation term of the supplementary component dictionary entry name is 'text', the  
4470 representation term MUST be removed from the supplementary component `xsd:attribute`  
4471 name.
- 4472 [R128] The attribute representing the supplementary component MUST be based on the appropriate XSD  
4473 built-in data type.
- 4474 [R129] For every core component type `xsd:complexType` definition a structured set of annotations  
4475 MAY be present in the following pattern:
- 4476 ○ UniqueID (required): The identifier that references the Core Component Type instance in a  
4477 unique and unambiguous way.
  - 4478 ○ Acronym (required): The abbreviation of the type of component. . In this case the value will  
4479 always be CCT.
  - 4480 ○ DictionaryEntryName (required): The official name of a Core Component Type.
  - 4481 ○ Version (required): An indication of the evolution over time of a Core Component Type  
4482 instance.
  - 4483 ○ Definition (required): The semantic meaning of a Core Component Type.
  - 4484 ○ PrimaryRepresentationTerm (required): The primary representation term of the Core  
4485 Component Type.
  - 4486 ○ PrimitiveType (required): The primitive data type of the Core Component Type.
  - 4487 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are  
4488 applicable to the Core Component Type.
- 4489 [R130] For every supplementary component `xsd:attribute` declaration a structured set of  
4490 annotations MAY be present in the following pattern:
- 4491 ○ UniqueID (optional): The identifier that references the Supplementary Component instance in  
4492 a unique and unambiguous way.
  - 4493 ○ Acronym (required): The abbreviation of the type of Supplementary Component. In this case  
4494 the value will always be SC.
  - 4495 ○ DictionaryEntryName (required): The official name of the Supplementary Component.
  - 4496 ○ Definition (required): The semantic meaning of the Supplementary Component.
  - 4497 ○ Cardinality (required): The cardinality of the Supplementary Component.
  - 4498 ○ ObjectClassTerm (required): The Object Class of the Supplementary Component.
  - 4499 ○ PropertyTerm (required): The Property Term of the Supplementary Component.
  - 4500 ○ PrimaryRepresentationTerm (required): The Primary Representation Term of the  
4501 Supplementary Component.
  - 4502 ○ PrimitiveType (required): The primitive data type of the Supplementary Component.
  - 4503 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are  
4504 applicable to the Supplementary Core Component.
- 4505 [R131] The Unqualified Data Type schema module namespace MUST be represented by the token `udt`.
- 4506 [R132] The `udt:UnqualifiedDataType` schema MUST only import the following schema  
4507 modules: – `ids:IdentifierList` schema modules – `clm:CodeList` schema modules

- 4508 [R133] An unqualified data type MUST be defined for each approved primary and secondary  
4509 representation terms identified in the CCTS Permissible Representation Terms table.
- 4510 [R134] The name of each unqualified data type MUST be the dictionary entry name of the primary or  
4511 secondary representation term, with the word 'Type' appended, the separators and spaces  
4512 removed and approved abbreviations applied.
- 4513 [R135] For every unqualified data type whose supplementary components map directly to the properties  
4514 of a XSD built-in data type, the unqualified data type MUST be defined as a named  
4515 **xsd:simpleType** in the **udt:UnqualifiedDataType** schema module.
- 4516 [R136] Every unqualified data type **xsd:simpleType** MUST contain one **xsd:restriction**  
4517 element. This **xsd:restriction** element MUST include an **xsd:base** attribute that  
4518 defines the specific XSD built-in data type required for the content component.
- 4519 [R137] For every unqualified data type whose supplementary components are not equivalent to the  
4520 properties of a XSD built-in data type, the unqualified data type MUST be defined as an  
4521 **xsd:complexType** in the **udt:UnqualifiedDataType** schema module.
- 4522 [R138] Every unqualified data type **xsd:complexType** definition MUST contain one  
4523 **xsd:simpleContent** element.
- 4524 [R139] Every unqualified data type **xsd:complexType** **xsd:simpleContent** element MUST  
4525 contain one **xsd:extension** element. This **xsd:extension** element must include an  
4526 **xsd:base** attribute that defines the specific XSD built-in data type required for the content  
4527 component.
- 4528 [R204] When a combination of the complex and simple types are necessary to support business  
4529 requirements, the element MUST be declared as an **xsd:complexType** with an  
4530 **xsd:choice** between elements declared as the two different alternatives.
- 4531 [R140] Within the unqualified data type **xsd:complexType** **xsd:extension** element an  
4532 **xsd:attribute** MUST be declared for each supplementary component pertaining to the  
4533 underlying CCT.
- 4534 [R141] Each supplementary component **xsd:attribute** name MUST be the supplementary  
4535 component name with the separators and spaces removed, and approved abbreviations and  
4536 acronyms applied.
- 4537 [R142] If the object class of the supplementary component dictionary entry name contains the name of  
4538 the representation term, the duplicated object class word or words MUST be removed from the  
4539 supplementary component **xsd:attribute** name.
- 4540 [R143] If the object class of the supplementary component dictionary entry name contains the term  
4541 'identification', the term 'identification' MUST be removed from the supplementary component  
4542 **xsd:attribute** name.
- 4543 [R144] If the representation term of the supplementary component dictionary entry name is 'text', the  
4544 representation term MUST be removed from the supplementary component **xsd:attribute**  
4545 name.
- 4546 [R145] If the representation term of the supplementary component is 'Code' and validation is required,  
4547 then the attribute representing this supplementary component MUST be based on the defined  
4548 **xsd:simpleType** of the appropriate external imported code list.
- 4549 [R146] If the representation term of the supplementary component is 'Identifier' and validation is  
4550 required, then the attribute representing this supplementary component MUST be based on the  
4551 defined **xsd:simpleType** of the appropriate external imported identifier list.
- 4552 [R147] If the representation term of the supplementary component is other than 'Code' or 'Identifier',  
4553 then the attribute representing this supplementary component MUST be based on the  
4554 appropriate XSD built-in data type.
- 4555 [R148] For every unqualified data type **xsd:complexType** or **xsd:simpleType** definition a  
4556 structured set of annotations MAY be present in the following pattern:
- 4557 ○ UniqueID (required): The identifier that references an Unqualified Data Type instance in a  
4558 unique and unambiguous way.
  - 4559 ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
4560 always be UDT.
  - 4561 ○ DictionaryEntryName (required): The official name of the Unqualified Data Type.
  - 4562 ○ Version (required): An indication of the evolution over time of the Unqualified Data Type  
4563 instance.



- 4564 ○ Definition (required): The semantic meaning of the Unqualified Data Type.
- 4565 ○ PrimaryRepresentationTerm (required): The primary representation term of the Unqualified
- 4566 Data Type.
- 4567 ○ PrimitiveType (required): The primitive data type of the Unqualified Data Type.
- 4568 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4569 applicable to the Unqualified Data Type.
- 4570 [R149] For every supplementary component **xsd:attribute** declaration a structured set of
- 4571 annotations MAY be present in the following pattern:
- 4572 ○ UniqueID (optional): The identifier that references a Supplementary Component instance in
- 4573 a unique and unambiguous way.
- 4574 ○ Acronym (required): The abbreviation of the type of component. In this case the value will
- 4575 always be SC.
- 4576 ○ Dictionary Entry Name (required): The official name of the Supplementary Component.
- 4577 ○ Definition (required): The semantic meaning of the Supplementary Component.
- 4578 ○ Cardinality (required): The cardinality of the Supplementary Component.
- 4579 ○ ObjectClassTerm (required): The Object Class of the Supplementary Component.
- 4580 ○ PropertyTerm (required): The Property Term of the Supplementary Component.
- 4581 ○ PrimaryRepresentationTerm (required): The Primary Representation Term of the
- 4582 Supplementary Component.
- 4583 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4584 applicable to the Supplementary Component.
- 4585 [R150] The Qualified Data Type schema module namespace MUST be represented by the token **qdt**.
- 4586 [R151] The **qdt:QualifiedDataType** schema module MUST import the
- 4587 **udt:UnqualifiedDataType schema module**.
- 4588 [R205] If a coupled design approach is used, then the **qdt:QualifiedDataType** schema module MUST
- 4589 import all code list and identifier scheme schemas used in the module..
- 4590 [R152] Where required to change facets of an existing unqualified data type, a new data type MUST be
- 4591 defined in the **qdt:QualifiedDataType** schema module.
- 4592 [R153] A qualified data type MUST be based on an unqualified or qualified data type and add some
- 4593 semantic and/or technical restriction to the base data type.
- 4594 [R154] The name of a qualified data type MUST be the name of its base unqualified or qualified data
- 4595 type with separators and spaces removed and with its qualifier term added.
- 4596 [R155] When a qualified data type is based on an unqualified data type that contains an
- 4597 **xsd:choice** element, then the qualified data type MUST be based on on or the other of the
- 4598 elements, but not both.
- 4599 [R156] Every qualified data type based on an unqualified or qualified data type **xsd:complexType**
- 4600 whose supplementary components do not map directly to the properties of a XSD built-in data
- 4601 type
- 4602 MUST be defined as a **xsd:complexType**
- 4603 MUST contain one **xsd:simpleContent** element
- 4604 MUST contain one **xsd:restriction** element
- 4605 MUST include the unqualified data type as its **xsd:base** attribute.
- 4606 [R157] Every qualified data type based on an unqualified or qualified data type
- 4607 **xsd:simpleType**
- 4608 MUST contain one **xsd:restriction** element
- 4609 MUST include the unqualified data type as its **xsd:base** attribute or if the facet
- 4610 restrictions can be achieved by use of a XSD built-in data type, then that XSD built-in
- 4611 data type may be used as the **xsd:base** attribute.
- 4612 [R158] Every qualified data type based on a single codelist or identifier list **xsd:simpleType** MUST
- 4613 contain one **xsd:restriction** element or **xsd:union** element. When using the
- 4614 **xsd:restriction** element, the **xsd:base** attribute MUST be set to the code list or identifier list
- 4615 schema module defined simple type with appropriate namespace qualification. When using the
- 4616 **xsd:union** element, the **xsd:member** type attribute MUST be set to the code list or identifier list
- 4617 schema module defined simple types with appropriate namespace qualification.
- 4618 [R159] Every qualified data type that has a choice of two or more code lists or identifier lists
- 4619 MUST be defined as an **xsd:complexType**

- 4620 MUST contain the `xsd:choice` element whose content model must consist of element  
 4621 references for the alternative code lists or identifier lists to be included with appropriate  
 4622 namespace qualification.
- 4623 [R160] The qualified data type `xsd:complexType` definition `xsd:simpleContent` element  
 4624 MUST only restrict attributes declared in its base type, or MUST only restrict facets  
 4625 equivalent to inherited supplementary components.
- 4626 [R161] Every qualified data type definition MAY contain a structured set of annotations in the  
 4627 following sequence and pattern:
- 4628 ○ UniqueID (required): The identifier that references a Qualified Data Type instance in a  
 4629 unique and unambiguous way.
  - 4630 ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
 4631 always be QDT.
  - 4632 ○ DictionaryEntryName (required): The official name of the Qualified Data Type.
  - 4633 ○ Version (required): An indication of the evolution over time of the Qualified Data Type  
 4634 instance.
  - 4635 ○ Definition (required): The semantic meaning of the Qualified Data Type.
  - 4636 ○ PrimaryRepresentationTerm (required): The Primary Representation Term of the Qualified  
 4637 Data Type.
  - 4638 ○ DataTypeQualifierTerm (required): A term that qualifies the Representation Term in order to  
 4639 differentiate it from its underlying Unqualified Data Type and other Qualified Data Type.
  - 4640 ○ PrimitiveType (required): The primitive data type of the Qualified Data Type.
  - 4641 ○ BusinessProcessContextValue (optional, repetitive): The business process context for this  
 4642 Qualified Data Type is associated.
  - 4643 ○ GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for  
 4644 this Qualified Data Type.
  - 4645 ○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this  
 4646 Qualified Data Type.
  - 4647 ○ ProductContextValue (optional, repetitive): The product context for this Qualified Data Type.
  - 4648 ○ IndustryContextValue (optional, repetitive): The industry context for this Qualified Data Type.
  - 4649 ○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this Qualified  
 4650 Data Type.
  - 4651 ○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this  
 4652 Qualified Data Type.
  - 4653 ○ SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for  
 4654 this Qualified Data Type.
  - 4655 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are  
 4656 applicable to the Qualified Data Type.
  - 4657 ○ Example (optional, repetitive): Example of a possible value of a Qualified Data Type.
- 4658 [R162] For every supplementary component `xsd:attribute` declaration a structured set of  
 4659 annotations MAY be present in the following pattern:
- 4660 ○ UniqueID (optional): The identifier that references a Supplementary Component of a Core  
 4661 Component Type instance in a unique and unambiguous way.
  - 4662 ○ Acronym (required): The abbreviation of the type of component. In this case the value will  
 4663 always be SC.
  - 4664 ○ DictionaryEntryName (required): The official name of a Supplementary Component.
  - 4665 ○ Definition (required): The semantic meaning of a Supplementary Component.
  - 4666 ○ Cardinality (required): Indication whether the Supplementary Component Property  
 4667 represents a not-applicable, optional, required and/or repetitive characteristic of the Core  
 4668 Component Type.
  - 4669 ○ ObjectClassTerm (required): The Object Class Term of the associated Supplementary  
 4670 Component.
  - 4671 ○ PropertyTerm (required): The Property Term of the associated Supplementary Component.
  - 4672 ○ PrimaryRepresentationTerm (required): The Primary Representation Term of the associated  
 4673 Supplementary Component.
  - 4674 ○ PrimitiveType (required): The Primitive Type of the associated Supplementary Component.
  - 4675 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are  
 4676 applicable to the Supplementary Component.
- 4677 [R163] Each UN/CEFACT maintained code list MUST be defined in its own schema module.

- 4678 [R164] Internal code list schema MUST NOT duplicate existing external code list schema when the  
4679 existing ones are available to be imported.
- 4680 [R165] The namespace names for code list schemas MUST have the following structure:  
4681 `urn:un:unece:uncefact:codelist:<status>:<Code List Agency`  
4682 `Identifier|Code List Agency Name Text>:<Code List Identification.`  
4683 `Identifier|Code List Name Text>:<Code List Version. Identifier>`  
4684 Where:  
4685 `codelist` = this token identifying the schema as a code list  
4686 `status` = a token identifying the standards status of this code list: `draft|standard`  
4687 `Code List Agency Identifier` = identifies the agency that manages a code list. The default  
4688 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.  
4689 `Code List Agency Name Text` = the name of the agency that maintains the code list.  
4690 `Code List Identification Identifier` = identifies a list of the respective corresponding codes. `listID`  
4691 is only unique within the agency that manages this code list. `Code List Name Text` = the  
4692 name of a list of codes.  
4693 `Code List Version Identifier` = identifies the version of a code list.
- 4694 [R166] This rule was combined with [R165].
- 4695 [R167] Each UN/CEFACT maintained code list schema module MUST be represented by a unique  
4696 token constructed as follows:  
4697 `clm[Qualified data type name]<Code List Agency Identifier|Code List`  
4698 `Agency Name Text><Code List Identification Identifier|Code List Name`  
4699 `Text>`  
4700 with any repeated words eliminated.
- 4701 [R168] The structure for schema location of code lists MUST be:  
4702 `../codelist/<status>:<Code List. Agency Identifier|Code List Agency`  
4703 `Name Text>:<Code List Identification Identifier|Code List Name`  
4704 `Text>_<Code List Version Identifier>.xsd`  
4705 Where:  
4706 `schematype` = a token identifying the type of schema module: `codelist`  
4707 `status` = the status of the schema as: `draft|standard`  
4708 `Code List Agency Identifier` = identifies the agency that manages a code list. The default  
4709 agencies used are those from DE 3055. `Code List Agency Name Text` = the name of the  
4710 agency that maintains the code list.  
4711 `Code List Identification Identifier` = identifies a list of the respective corresponding codes.  
4712 `listID` is only unique within the agency that manages this code list.  
4713 `Code List Name Text` = the name of a list of codes.  
4714 `Code List Version Identifier` = identifies the version of a code list.
- 4715 [R169] Each `xsd:schemaLocation` attribute declaration of a code list MUST contain a resolvable  
4716 URL, and if an absolute path is used, it MUST also be persistent.
- 4717 [R170] This rule has been removed.
- 4718 [R171] Code List schema modules MUST not import or include any other schema modules.
- 4719 [R172] Within each code list module one, and only one, named `xsd:simpleType` MUST be defined for  
4720 the content component.
- 4721 [R173] The name of the `xsd:simpleType` MUST be the name of code list root element with the  
4722 word 'ContentType' appended.
- 4723 [R174] The `xsd:restriction` element base attribute value MUST be set to `xsd:token`.
- 4724 [R175] Each code in the code list MUST be expressed as an `xsd:enumeration`, where the  
4725 `xsd:value` for the enumeration is the actual code value.
- 4726 [R176] For each code list a single root element MUST be globally declared.
- 4727 [R177] The name of the code list root element MUST be the name of the code list following the  
4728 naming rules as defined in section 5.3.
- 4729 [R178] The code list root element MUST be of a type representing the actual list of code values.
- 4730 [R179] Each code list `xsd:enumeration` MAY contain a structured set of annotations in the  
4731 following sequence and pattern:  
4732
  - Name (required): The name of the code.
  - Description (optional): Descriptive information concerning the code.  
4733

- 4734 [R180] Internal identifier lists schema MUST NOT duplicate existing external identifier list schema  
4735 when the existing ones are available to be imported.
- 4736 [R181] Each UN/CEFACT maintained identifier list MUST be defined in its own schema module.
- 4737 [R182] The names for namespaces MUST have the following structure:  
4738 `urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme.`  
4739 `Agency Identifier|Identifier Scheme Agency Name Text>:<Identifier`  
4740 `Scheme Identifier|Identifier Scheme Name Text>:<Identifier Scheme`  
4741 `Version Identifier>`  
4742 Where:  
4743 status = the token identifying the publication status of this identifier scheme schema =  
4744 `draft|standard`  
4745 identifierlist = this token identifying the schema as an identifier scheme  
4746 Identifier Scheme Agency Identifier = the identification of the agency that maintains the  
4747 identification scheme.  
4748 Identifier Scheme Agency Name. Text = the name of the agency that maintains the  
4749 identification list.  
4750 Identifier Scheme Identifier = the identification of the identification scheme.  
4751 Identifier Scheme Name. Text = the name of the identification scheme.  
4752 Identifier Scheme Version. Identifier = the version of the identification scheme.
- 4753 [R183] This rule was combined with [R182].
- 4754 [R184] Each UN/CEFACT maintained identifier list schema module MUST be represented by a  
4755 unique token constructed as follows:  
4756 `ids[Qualified data type name]<Identification Scheme Agency`  
4757 `Identifier><Identification Scheme Identifier>`  
4758 with any repeated words eliminated.
- 4759 [R185] The structure for schema location of identifier lists MUST be:  
4760 `[./identifierlist/<status>:<Identifier Scheme Agency Identifier|Identifier`  
4761 `Scheme Agency Name Text>/< Identifier Scheme Identifier|Identifier`  
4762 `Scheme Name Text>_< Identifier Scheme Version Identifier>.xsd`  
4763 Where:  
4764 schematype = a token identifying the type of schema module: `identifierlist`  
4765 status = the status of the schema as: `draft|standard`  
4766 Identifier Scheme. Agency Identifier = the identification of the agency that maintains the  
4767 identification scheme.  
4768 Identifier Scheme. Agency Name. Text = the name of the agency that maintains the  
4769 identification scheme.  
4770 Identifier Scheme. Identifier = the identification of the identification scheme.  
4771 Identifier Scheme. Name. Text = the name of the identification scheme.  
4772 Identifier Scheme. Version. Identifier = the version of the identification scheme.
- 4773 [R186] Each `xsd:schemaLocation` attribute declaration of an identifier list schema MUST contain a  
4774 resolvable URL, and if an absolute path is used, it MUST also be persistent.
- 4775 [R187] This rule has been removed.
- 4776 [R188] Identifier list schema modules MUST NOT import or include any other schema modules.
- 4777 [R189] Within each identifier list schema module one, and only one, named `xsd:simpleType` MUST  
4778 be defined for the content component.
- 4779 [R190] The name of the `xsd:simpleType` MUST be the name of the identifier list root element  
4780 with the word 'ContentType' appended.
- 4781 [R191] The `xsd:restriction` element base attribute value MUST be set to `xsd:token`.
- 4782 [R192] Each identifier in the identifier list MUST be expressed as an `xsd:enumeration`, where the  
4783 `xsd:value` for the enumeration is the actual identifier value.
- 4784 [R193] Facets other than `xsd:enumeration` MUST NOT be used in the identifier list schema  
4785 module.
- 4786 [R194] For each identifier list a single root element MUST be globally declared.
- 4787 [R195] The name of the identifier list root element MUST be the name of the identifier list following the  
4788 naming rules as defined in section 5.3.

- 4789 [R196] The identifier list root element MUST be of a type representing the actual list of identifier  
4790 values.
- 4791 [R197] Each **xsd:enumeration** MAY contain a structured set of annotations in the following  
4792 sequence and pattern:
- 4793 ○ Name (required): The name of the identifier.
  - 4794 ○ Description (optional): Descriptive information concerning the identifier.
- 4795 [R198] All UN/CEFACT XML MUST be instantiated using UTF. UTF-8 should be used as the  
4796 preferred encoding. If UTF-8 is not used, UTF-16 MUST be used.
- 4797 [R199] The **xsi** prefix MUST be used where appropriate for referencing **xsd:schemaLocation** and  
4798 **xsd:noNamespaceLocation** attributes in instance documents.
- 4799 [R200] UN/CEFACT conformant instance documents MUST NOT contain an element devoid of  
4800 content.
- 4801 [R201] The **xsi:nil** attribute MUST NOT appear in any conforming instance.
- 4802 [R202] The **xsi:type** attribute MUST NOT be used
- 4803



## Appendix G: Glossary

4804

4805 *Aggregate Business Information Entity (ABIE)* – A collection of related pieces of business information that  
4806 together convey a distinct business meaning in a specific *Business Context*. Expressed in modelling terms,  
4807 it is the representation of an *Object Class*, in a specific *Business Context*.

4808 *Aggregate Core Component - (ACC)* – A collection of related pieces of business information that together  
4809 convey a distinct business meaning, independent of any specific *Business Context*. Expressed in modelling  
4810 terms, it is the representation of an *Object Class*, independent of any specific *Business Context*.

4811 *Aggregation* – An *Aggregation* is a special form of *Association* that specifies a whole-part relationship  
4812 between the aggregate (whole) and a component part.

4813 *Association Business Information Entity (ASBIE)* - A *Business Information Entity* that represents a complex  
4814 business characteristic of a specific *Object Class* in a specific *Business Context*. It has a unique *Business*  
4815 *Semantic* definition. An *Association Business Information Entity* represents an *Association Business*  
4816 *Information Entity Property* and is therefore associated to an *Aggregate Business Information Entity*, which  
4817 describes its structure. An *Association Business Information Entity* is derived from an *Association Core*  
4818 *Component*.

4819 *Association Business Information Entity Property* - A *Business Information Entity Property* for which the  
4820 permissible values are expressed as a complex structure, represented by an *Aggregate Business*  
4821 *Information Entity*.

4822 *Association Core Component (ASCC)* - A *Core Component* which constitutes a complex business  
4823 characteristic of a specific *Aggregate Core Component* that represents an *Object Class*. It has a unique  
4824 *Business Semantic* definition. An *Association Core Component* represents an *Association Core*  
4825 *Component Property* and is associated to an *Aggregate Core Component*, which describes its structure.

4826 *Association Core Component Property* – A *Core Component Property* for which the permissible values  
4827 are expressed as a complex structure, represented by an *Aggregate Core Component*.

4828 *Association Type* – The association type of the *Association Business Information Entity*.

4829 *Attribute* – A named value or relationship that exists for some or all instances of some entity and is  
4830 directly associated with that instance.

4831 *Basic Business Information Entity (BBIE)* – A *Business Information Entity* that represents a singular  
4832 business characteristic of a specific *Object Class* in a specific *Business Context*. It has a unique *Business*  
4833 *Semantic* definition. A *Basic Business Information Entity* represents a *Basic Business Information*  
4834 *Entity Property* and is therefore linked to a *Data Type*, which describes its values. A *Basic Business Information*  
4835 *Entity* is derived from a *Basic Core Component*.

4836 *Basic Business Information Entity Property* – A *Business Information Entity Property* for which the  
4837 permissible values are expressed by simple values, represented by a *Data Type*.

4838 *Basic Core Component (BCC)* – A *Core Component* which constitutes a singular business characteristic of  
4839 a specific *Aggregate Core Component* that represents a *Object Class*. It has a unique *Business Semantic*  
4840 definition. A *Basic Core Component* represents a *Basic Core Component Property* and is therefore of a  
4841 *Data Type*, which defines its set of values. *Basic Core Components* function as the properties of  
4842 *Aggregate Core Components*.

4843 *Basic Core Component (CC) Property* – A *Core Component Property* for which the permissible values are  
4844 expressed by simple values, represented by a *Data Type*.

4845 *Business Context* – The formal description of a specific business circumstance as identified by the values of  
4846 a set of *Context Categories*, allowing different business circumstances to be uniquely distinguished.

4847 *Business Information Entity (BIE)* – A piece of business data or a group of pieces of business data with a unique  
4848 *Business Semantic* definition. A *Business Information Entity* can be a *Basic Business Information Entity* (BBIE),  
4849 an *Association Business Information Entity* (ASBIE), or an *Aggregate Business Information*  
4850 *Entity* (ABIE).

4851 *Business Information Entity (BIE) Property* – A business characteristic belonging to the *Object Class* in its  
4852 specific *Business Context* that is represented by an *Aggregate Business Information Entity*.

4853 *Business Libraries* – A collection of approved process models specific to a line of business (e.g.,  
4854 shipping, insurance).

4855 *Business Process* – The *Business Process* as described using the UN/CEFACT Modelling Methodology.

4856 *Business Process Context* – The *Business Process* name(s) as described using an appropriate list  
4857 of relevant business processes.

4858 *Business Process Role Context* – The actor(s) conducting a particular Business Process.

4859 *Business Semantic(s)* – A precise meaning of words from a business perspective.

4860 *Business Term* – This is a synonym under which the *Core Component* or *Business Information Entity* is  
4861 commonly known and used in the business. A *Core Component* or *Business Information Entity* may have  
4862 several *Business Terms* or synonyms.

4863 *Cardinality* – An indication whether a characteristic is optional, mandatory and/or repetitive.

4864 *CCL* – see *Core Component Library*.

4865 *Classification Scheme* – This is an officially supported scheme to describe a given *Context Category*.

4866 *Composition* – A form of aggregation which requires that a part instance be included in at most one  
4867 composite at a time, and that the composite object is responsible for the creation and destruction of the  
4868 parts. *Composition* may be recursive.

4869 *Content Component* – Defines the *Primitive Type* used to express the content of a *Core Component Type*.

4870 *Content Component Restrictions* – The formal definition of a format restriction that applies to the possible  
4871 values of a *Content Component*.

4872 *Context* – Defines the circumstances in which a *Business Process* may be used. This is specified by a set of  
4873 *Context Categories* known as *Business Context*.

4874 *Context Category* – A group of one or more related values used to express a characteristic of a business  
4875 circumstance.

4876 *Controlled Vocabulary* – A supplemental vocabulary used to define potentially ambiguous words or  
4877 *Business Terms*. This ensures that every word within any of the core component names and definitions is  
4878 used consistently, unambiguously and accurately.

4879 *Core Component (CC)* – A building block for the creation of a semantically correct and meaningful  
4880 information exchange package. It contains only the information pieces necessary to describe a specific  
4881 concept.

4882 *Core Component Library* – The *Core Component Library* will contain all the *Core Component Types*, *Basic*  
4883 *Core Components*, *Aggregate Core Components*, *Basic Business Information Entities*, *Aggregate*  
4884 *Business Information Entities*, and *Data Types*.

4885 *Core Component Property* – A business characteristic belonging to the *Object Class* represented by an  
4886 *Aggregate Core Component*.

4887 *Core Component Type (CCT)* – A *Core Component*, which consists of one and only one *Content*  
4888 *Component*, that carries the actual content plus one or more *Supplementary Components* giving an  
4889 essential extra definition to the *Content Component*. *Core Component Types* do not have *Business*  
4890 *Semantics*.

4891 *Data Type* – Defines the set of valid values that can be used for a particular *Basic Core Component*  
4892 *Property* or *Basic Business Information Entity Property*. It is defined by specifying restrictions on the *Core*  
4893 *Component Type* that forms the basis of the *Data Type*.

4894 *Decoupling* – The term “*Decoupling*” refers to *decoupling of unqualified data type schema from the*  
4895 *data type catalogue (CCT) and/or decoupling a particular qualified data type from a set of value*  
4896 *enumerations*.

4897 *Definition* – This is the unique semantic meaning of a *Core Component*, *Business Information Entity*,  
4898 *Business Context* or *Data Type*.



4899 *Dictionary Entry Name* – This is the unique official name of a *Core Component*, *Business Information Entity*,  
4900 *Business Context* or *Data Type* in the library.

4901 *Geopolitical Context* – A combination of political and geographic factors influencing or delineating a  
4902 country or region.

4903 *Industry Classification Context* – Semantic influences related to the industry or industries of the trading  
4904 partners (e.g., product identification schemes used in different industries).

4905 *Information Entity* – A reusable semantic building block for the exchange of business-related information.

4906 *Lower-Camel-Case (LCC)* – a style that capitalizes the first character of each word except the first word  
4907 and compounds the name.

4908 *Naming Convention* – The set of rules that together comprise how the *Dictionary Entry Name* for *Core*  
4909 *Components* and *Business Information Entities* are constructed.

4910 *Object Class* – The logical data grouping (in a logical data model) to which a data element belongs  
4911 (ISO11179). The *Object Class* is the part of a *Core Component's Dictionary Entry Name* that represents  
4912 an activity or object in a specific *Context*.

4913 *Object Class Term* – A component of the name of a *Core Component* or *Business Information Entity* which  
4914 represents the *Object Class* to which it belongs.

4915 *Official Constraints Context* – Legal and governmental influences on semantics (e.g. hazardous materials  
4916 information required by law when shipping goods).

4917 *Primitive Type* – Used for the representation of a value. Possible values are String, Decimal, Integer,  
4918 Boolean, Date and Binary.

4919 *Product Classification Context* – Factors influencing semantics that are the result of the goods or services  
4920 being exchanged, handled, or paid for, etc. (e.g. the buying of consulting services as opposed to materials)

4921 *Property* – A peculiarity common to all members of an *Object Class*.

4922 *Property Term* – A semantically meaningful name for the characteristic of the *Object Class* that is  
4923 represented by the *Core Component Property*. It shall serve as basis for the *Dictionary Entry Name* of the  
4924 *Basic* and *Association Core Components* that represents this *Core Component Property*.

4925 *Qualifier Term* – A word or group of words that help define and differentiate an item (e.g. a *Business*  
4926 *Information Entity* or a *Data Type*) from its associated items (e.g. from a *Core Component*, a *Core*  
4927 *Component Type*, another *Business Information Entity* or another *Data Type*).

4928 *Registry Class* – The formal definition of all the information necessary to be recorded in the Registry  
4929 about a *Core Component*, a *Business Information Entity*, a *Data Type* or a *Business Context*.

4930 *Representation Term* – The type of valid values for a *Basic Core Component* or *Business Information Entity*.

4931 *Supplementary Component* – Gives additional meaning to the *Content Component* in the *Core Component*  
4932 *Type*.

4933 *Supplementary Component Restrictions* – The formal definition of a format restriction that applies to the  
4934 possible values of a *Supplementary Component*.

4935 *Supporting Role Context* – Semantic influences related to non-partner roles (e.g., data required by a third-  
4936 party shipper in an order response going from seller to buyer.)

4937 *Syntax Binding* – The process of expressing a *Business Information Entity* in a specific syntax.

4938 *System Capabilities Context* – This *Context category* exists to capture the limitations of systems (e.g. an  
4939 existing back office can only support an address in a certain form).

4940 *UMM Information Entity* – A *UMM Information Entity* realizes structured business information that is  
4941 exchanged by partner roles performing activities in a business transaction. Information entities include or  
4942 reference other information entities through associations.”

4943 *Unique Identifier* – The identifier that references a *Registry Class* instance in a universally unique and  
4944 unambiguous way.

- 4945 *Upper-Camel-Case (UCC)* – a style that capitalizes the first character of each word and compounds the  
4946 name.
- 4947 *Usage Rules* – *Usage Rules* describe how and/or when to use the *Registry Class*.
- 4948 *User Community* – A *User Community* is a group of practitioners, with a publicised contact address, who  
4949 may define *Context* profiles relevant to their area of business. Users within the community do not create,  
4950 define or manage their individual *Context* needs but conform to the community’s standard. Such a  
4951 community should liase closely with other communities and with general standards-making bodies to avoid  
4952 overlapping work. A community may be as small as two consenting organisations.
- 4953 *Version* – An indication of the evolution over time of an instance of a *Core Component*, *Data Type*, *Business*  
4954 *Context*, or *Business Information Entity*.
- 4955 *XML schema* – A Recommendation of the World Wide Web Consortium (W3C), which specifies how to  
4956 formally describe the elements in an Extensible Markup Language (XML) document. This description can  
4957 be used to verify that each item of content in a document adheres to the description of the element in  
4958 which the content is to be placed.
- 4959

## 4960 **Intellectual Property Disclaimer**

4961 ECE draws attention to the possibility that the practice or implementation of its outputs (which include but are  
4962 not limited to Recommendations, norms, standards, guidelines and technical specifications) may involve the  
4963 use of a claimed intellectual property right.

4964 Each output is based on the contributions of participants in the UN/CEFACT process, who have agreed to  
4965 waive enforcement of their intellectual property rights pursuant to the UN/CEFACT IPR Policy (document  
4966 ECE/TRADE/C/CEFACT/2010/20/Rev.2 available at [http://www.unece.org/cefact/cf\\_docs.html](http://www.unece.org/cefact/cf_docs.html) or from the  
4967 ECE secretariat). ECE takes no position concerning the evidence, validity or applicability of any claimed  
4968 intellectual property right or any other right that might be claimed by any third parties related to the  
4969 implementation of its outputs. ECE makes no representation that it has made any investigation or effort to  
4970 evaluate any such rights.

4971 Implementers of UN/CEFACT outputs are cautioned that any third-party intellectual property rights claims  
4972 related to their use of a UN/CEFACT output will be their responsibility and are urged to ensure that their use  
4973 of UN/CEFACT outputs does not infringe on an intellectual property right of a third party.

4974 ECE does not accept any liability for any possible infringement of a claimed intellectual property right or any  
4975 other right that might be claimed to relate to the implementation of any of its outputs.