

1

2

3

XML Naming and Design Rules For CCTS 2.01

4

5

6

Version 2.1

27 May 2014

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 Hisanao Sugamata

28 *Contributors to Previous Versions*

29 Mark Crawford
30 Paula Heilig
31 Gunter Stuhec
32 Garret Minikawa
33 Hisanao Sugamata
34 Frank Lin
35 K.K. Suen
36 Luc Mouchot
37 Thomas Bikeev
38 Jostein Frømyr
39 Sue Probert
40 Alain Dechamps
41 Michael Dill

42

3 Table of Contents

| | | | |
|----|-------|--|----|
| 43 | 1 | Status of this Document | 2 |
| 44 | 2 | UN/CEFACT XML Naming and Design Rules Project Team Participants..... | 3 |
| 45 | 4 | Introduction..... | 9 |
| 46 | 4.1 | Scope and Focus | 9 |
| 47 | 4.2 | Audience..... | 9 |
| 48 | 4.3 | Structure of this Specification | 9 |
| 49 | 4.4 | Terminology and Notation | 10 |
| 50 | 4.5 | Related Documents | 10 |
| 51 | 4.6 | Conformance | 10 |
| 52 | 4.7 | Guiding Principles | 10 |
| 53 | 5 | General XML Construct..... | 12 |
| 54 | 5.1 | Overall Schema Structure..... | 12 |
| 55 | 5.2 | Relationship to CCTS..... | 12 |
| 56 | 5.2.1 | CCTS | 12 |
| 57 | 5.2.2 | Business Information Entities | 13 |
| 58 | 5.2.3 | The XML Constructs..... | 14 |
| 59 | 5.3 | Naming and Modelling Constraints..... | 16 |
| 60 | 5.3.1 | Element Naming Conventions | 18 |
| 61 | 5.4 | Reusability Scheme (Informative) | 18 |
| 62 | 5.5 | Modularity Model..... | 19 |
| 63 | 5.5.1 | Root Schema..... | 21 |
| 64 | 5.5.2 | Internal Schema..... | 22 |
| 65 | 5.5.3 | External Schema | 22 |
| 66 | 5.6 | Namespace Scheme | 25 |
| 67 | 5.6.1 | Namespace Scheme..... | 26 |
| 68 | 5.6.2 | Declaring Namespace | 26 |
| 69 | 5.6.3 | Namespace Persistence..... | 27 |
| 70 | 5.6.4 | Namespace Uniform Resource Identifiers..... | 27 |
| 71 | 5.6.5 | Namespace Constraint | 28 |
| 72 | 5.6.6 | UN/CEFACT XSD Namespace Schema Tokens | 28 |
| 73 | 5.7 | Schema Location..... | 28 |
| 74 | 5.8 | Versioning..... | 29 |
| 75 | 5.8.1 | Major Versions..... | 29 |

| | | |
|-----|--|----|
| 77 | 5.8.2 Minor Versions | 29 |
| 78 | 6 General XML Schema Language Conventions | 31 |
| 79 | 6.1 Schema Construct..... | 31 |
| 80 | 6.1.1 Constraints on Schema Construction | 31 |
| 81 | 6.2 Attribute and Element Declarations..... | 31 |
| 82 | 6.2.1 Attributes..... | 31 |
| 83 | 6.2.2 Elements..... | 32 |
| 84 | 6.3 Type Declarations | 32 |
| 85 | 6.3.1 Usage of Types..... | 32 |
| 86 | 6.3.2 Simple Type Definitions..... | 33 |
| 87 | 6.3.3 Complex Type Definitions..... | 33 |
| 88 | 6.4 User of XSD Extension and Restriction | 33 |
| 89 | 6.4.1 Extension | 34 |
| 90 | 6.4.2 Restriction..... | 34 |
| 91 | 6.5 Annotation..... | 34 |
| 92 | 6.5.1 Documentation..... | 34 |
| 93 | 7 XML Schema Modules | 38 |
| 94 | 7.1 Root Schema..... | 38 |
| 95 | 7.1.1 Schema Construct..... | 38 |
| 96 | 7.1.2 Namespace Scheme..... | 39 |
| 97 | 7.1.3 Imports and Includes | 39 |
| 98 | 7.1.4 Root Element Declaration | 40 |
| 99 | 7.1.5 Type Definitions..... | 40 |
| 100 | 7.1.6 Annotations | 41 |
| 101 | 7.2 Internal Schema..... | 41 |
| 102 | 7.2.1 Schema Construct..... | 41 |
| 103 | 7.2.2 Namespace Scheme..... | 41 |
| 104 | 7.2.3 Imports and Includes | 41 |
| 105 | 7.3 Reusable Aggregate Business Information Entity Schema | 42 |
| 106 | 7.3.1 Schema Construct..... | 42 |
| 107 | 7.3.2 Namespace Scheme..... | 42 |
| 108 | 7.3.3 Imports and Includes | 42 |
| 109 | 7.3.4 Type Declarations | 43 |
| 110 | 7.3.5 Element Declarations and References..... | 45 |

| | | |
|-----|--|----|
| 111 | 7.3.6 Annotation..... | 46 |
| 112 | 7.4 Core Component Type..... | 49 |
| 113 | 7.4.1 Use of Core Component Type Module | 49 |
| 114 | 7.4.2 Schema Construct..... | 49 |
| 115 | 7.4.3 Namespace Scheme..... | 50 |
| 116 | 7.4.4 Imports and Includes | 50 |
| 117 | 7.4.5 Type Definitions..... | 50 |
| 118 | 7.4.6 Attribute Declarations | 51 |
| 119 | 7.4.7 Extension and Restriction | 51 |
| 120 | 7.4.8 Annotation..... | 51 |
| 121 | 7.5 Unqualified Data Type | 52 |
| 122 | 7.5.1 Use of Unqualified Data Type Module | 52 |
| 123 | 7.5.2 Schema Construct..... | 53 |
| 124 | 7.5.3 Namespace Scheme..... | 53 |
| 125 | 7.5.4 Imports and Includes | 53 |
| 126 | 7.5.5 Type Definitions..... | 54 |
| 127 | 7.5.6 Attribute Declarations | 54 |
| 128 | 7.5.7 Extension and Restriction | 57 |
| 129 | 7.5.8 Annotation..... | 57 |
| 130 | 7.6 Qualified Data Type | 58 |
| 131 | 7.6.1 Use of Qualified Data Type Schema Module..... | 58 |
| 132 | 7.6.2 Schema Construct..... | 58 |
| 133 | 7.6.3 Namespace Scheme..... | 59 |
| 134 | 7.6.4 Imports and Includes | 59 |
| 135 | 7.6.5 Type Definitions..... | 59 |
| 136 | 7.6.6 Attribute and Element Declarations..... | 61 |
| 137 | 7.6.7 Annotation..... | 62 |
| 138 | 7.7 Code Lists..... | 63 |
| 139 | 7.7.1 Schema Construct..... | 63 |
| 140 | 7.7.2 Namespace Name for Code Lists..... | 64 |
| 141 | 7.7.3 UN/CEFACT XSD Schema Namespace Token for Code Lists..... | 65 |
| 142 | 7.7.4 Schema Location..... | 66 |
| 143 | 7.7.5 Imports and Includes | 67 |
| 144 | 7.7.6 Type Definitions..... | 67 |

| | | | |
|-----|------------|---|----|
| 145 | 7.7.7 | Element and Attribute Declarations..... | 68 |
| 146 | 7.7.8 | Extension and Restriction | 68 |
| 147 | 7.7.9 | Annotation..... | 68 |
| 148 | 7.8 | Identifier List Schema | 69 |
| 149 | 7.8.1 | Schema Construct..... | 69 |
| 150 | 7.8.2 | Namespace Name For Identifier List Schema..... | 70 |
| 151 | 7.8.3 | UN/CEFACT XSD Namespace Token for Identifier List Schema | 71 |
| 152 | 7.8.4 | Schema Location..... | 71 |
| 153 | 7.8.5 | Imports and Includes | 72 |
| 154 | 7.8.6 | Type Definitions..... | 72 |
| 155 | 7.8.7 | Attribute and Element Declarations..... | 73 |
| 156 | 7.8.8 | Extension and Restriction | 73 |
| 157 | 7.8.9 | Annotation..... | 74 |
| 158 | 8 | XML Instance Documents..... | 75 |
| 159 | 8.1 | Character Encoding | 75 |
| 160 | 8.2 | xsi:schemaLocation..... | 75 |
| 161 | 8.3 | Empty Content..... | 75 |
| 162 | 8.4 | xsi:type..... | 75 |
| 163 | Appendix A | Related Documents | 76 |
| 164 | Appendix B | Overall Structure..... | 77 |
| 165 | B.1 | XML Declaration | 77 |
| 166 | B.2 | Schema Module Identification and Intellectual Property Disclaimer | 77 |
| 167 | B.3 | Schema Start Tag | 77 |
| 168 | B.4 | Includes..... | 78 |
| 169 | B.5 | Imports | 78 |
| 170 | B.6 | Root Element..... | 79 |
| 171 | B.7 | Type Definitions..... | 80 |
| 172 | Appendix C | BPS Approved Acronyms and Abbreviations | 83 |
| 173 | Appendix D | Common Use Cases for Code Lists and Identifier Lists..... | 84 |
| 174 | D.1 | The Use of Code Lists within XML Schemas | 84 |
| 175 | D.1.1 | Referencing a Predefined Standard Code List in and Unqualified Data Type | 85 |
| 176 | D.1.2 | Referencing Any Code List Using the Unqualified Data Type udt:CodeType | 86 |
| 177 | D.1.3 | Referencing a Predefined Code List by Declaring a Specific Qualified Data Type..... | 86 |
| 178 | D.1.4 | Choosing or Combining Values from Different Code Lists | 87 |

| | | | |
|-----|--|--|-----|
| 179 | D.1.5 | Restricting Allowed Code Values | 88 |
| 180 | D.2 | The Use of Identifier Schemes within XML Schemas | 89 |
| 181 | Appendix E | Annotation Templates | 90 |
| 182 | Appendix F | Naming and Design Rules Checklist..... | 94 |
| 183 | Appendix G: | Glossary | 108 |
| 184 | Intellectual Property Disclaimer | | 112 |
| 185 | | | |

186 4 Introduction

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

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

193 4.1 Scope and Focus

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

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

202 This version was amended from the original to correct certain errors in the original text, as well as
203 to fulfill the following goals: (1) decoupling the unqualified data type schema from this
204 specification, to allow for easier maintenance of the data type catalogue, (2) to enable relative
205 path names, making implementation of the schemas easier, and (3) incorporating the Core
206 Component Business Document Assembly specification, which did not exist at the time the first
207 version was created.

208 4.2 Audience

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

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

220 4.3 Structure of this Specification

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

- 223 • Section 4 provides general information about the document itself as well as normative
224 statements in respect to conformance, and guiding principles applied in developing this
225 specification.
- 226 • Section 5 provides information on this specification's dependency and relationship to
227 CCTS. Furthermore, this section describes the approach taken to modularity in order to
228 maximize the reuse of business information expressed as XML schema components and
229 the general naming conventions applied. (Normative, except for Section 5.4, which is
230 Informative)
- 231 • Section 6 provides the general conventions applied with respect to the use of the XML
232 schema language. (Normative)
- 233 • Section 7 provides detailed rules applicable to each of the schema modules defined
234 by the modularity approach. (Normative)

- 235 • Section 8 provides guidelines and rules related to XML instance documents. (Normative)

236 The document also contains the following Appendices:

- 237 • Appendix A Related Documents (Informative)
- 238 • Appendix B Overall Structure (Normative)
- 239 • Appendix C BPS Approved Acronyms and Abbreviations (Normative)
- 240 • Appendix D Use cases for code lists and identifier lists. (Informative)
- 241 • Appendix E Annotation Templates (Informative)
- 242 • Appendix F Naming and Design Rules List (Normative)
- 243 • Appendix G Glossary (Informative)

244 4.4 Terminology and Notation

245 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
246 RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted
247 as described in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2119.¹

248 Wherever xsd: appears this refers to a construct taken from the W3C XML schema specification.

249 Wherever ccts: appears this refers to a construct taken from the CCTS.

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

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

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

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

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

- 257 • [] = optional
- 258 • < > = Variable
- 259 • | = choice

260 4.5 Related Documents

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

262 4.6 Conformance

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

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

267 4.7 Guiding Principles

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

- 270 ○ Relationship to UMM – UN/CEFACT XML Schema Definition Language (XSD) Schema will
271 be based on UMM metamodel adherent Business Process Models.
- 272 ○ Relationship to Information Models – UN/CEFACT XSD Schema will be based on
273 information models developed in accordance with the UN/CEFACT – *Core Components*
274 *Technical Specification*.

¹ 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 ○ Schema Creation– UN/CEFACT XML design rules will support schema creation
276 through handcrafting as well as automatic generation.
277 ○ Interchange and Application Use – UN/CEFACT XSD Schema and instance documents
278 are intended for business-to-business and application-to-application use.
279 ○ Tool Use and Support - The design of UN/CEFACT XSD Schema will not make any
280 assumptions about sophisticated tools for creation, management, storage, or presentation
281 being available.
282 ○ Legibility - UN/CEFACT XML instance documents should be intuitive and reasonably clear in
283 the context for which they are designed.
284 ○ Schema Features - The design of UN/CEFACT XSD Schema should use the most
285 commonly supported features of W3C XSD Schema.
286 ○ Technical Specifications – UN/CEFACT XML Naming and Design Rules will be based on
287 Technical Specifications holding the equivalent of W3C recommended status.
288 ○ Schema Specification – UN/CEFACT XML Naming and Design rules will be fully conformant with
289 W3C XML Schema Definition Language.
290 ○ Interoperability - The number of ways to express the same information in a UN/CEFACT
291 XSD Schema and UN/CEFACT XML instance document is to be kept as close to one as
292 possible.
293 ○ Maintenance – The design of UN/CEFACT XSD Schema must facilitate maintenance.
294 ○ Context Sensitivity - The design of UN/CEFACT XSD Schema must ensure that context-
295 sensitive document types are not precluded.
296 ○ Relationship to Other Namespaces - UN/CEFACT XML design rules will be cautious about
297 making dependencies on other namespaces.
298 ○ Legacy formats - UN/CEFACT XML Naming and Design Rules are not responsible for
299 sustaining legacy formats.
300

301 5 General XML Construct

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

- 303 ○ Overall Schema Structure
- 304 ○ Relationship to CCTS
- 305 ○ Naming and Modelling Constraints
- 306 ○ Reusability Scheme
- 307 ○ Modularity Model
- 308 ○ Namespace Scheme
- 309 ○ Schema Location
- 310 ○ Versioning Scheme

312 5.1 Overall Schema Structure

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

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

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

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

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

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

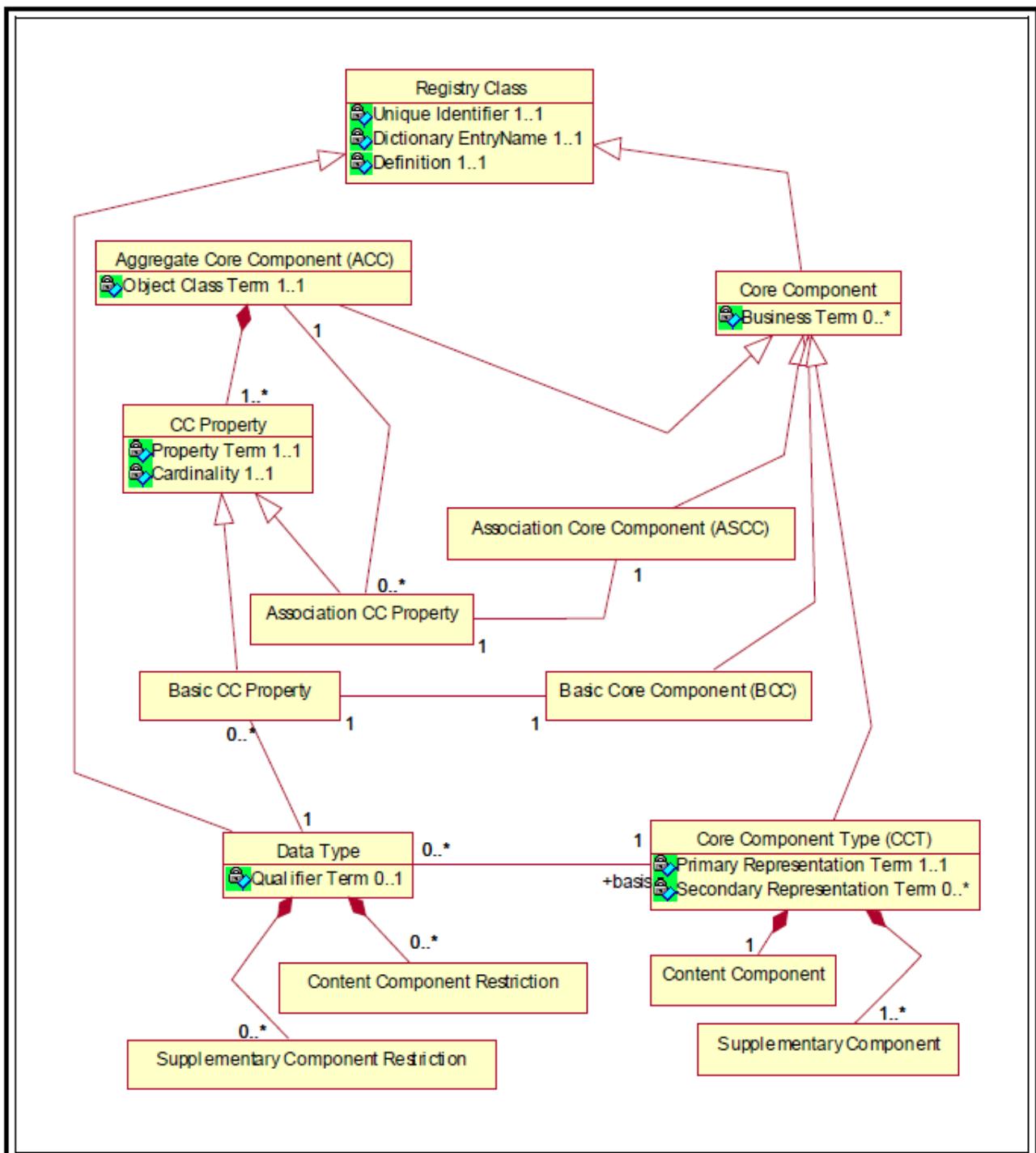
329 5.2 Relationship to CCTS

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

332 5.2.1 CCTS

333 CCTS defines context neutral and context specific information building blocks. Context neutral
334 information components are defined as Core Components (`ccts:CoreComponents`). Context
335 neutral `ccts:CoreComponents` are defined in CCTS as “A building block for the creation of a
336 semantically correct and meaningful information exchange package. It contains only the
337 information pieces necessary to describe a specific concept.”² Figure 5-1 illustrates the various
338 pieces of the overall `ccts:CoreComponents` metamodel.

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



339

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

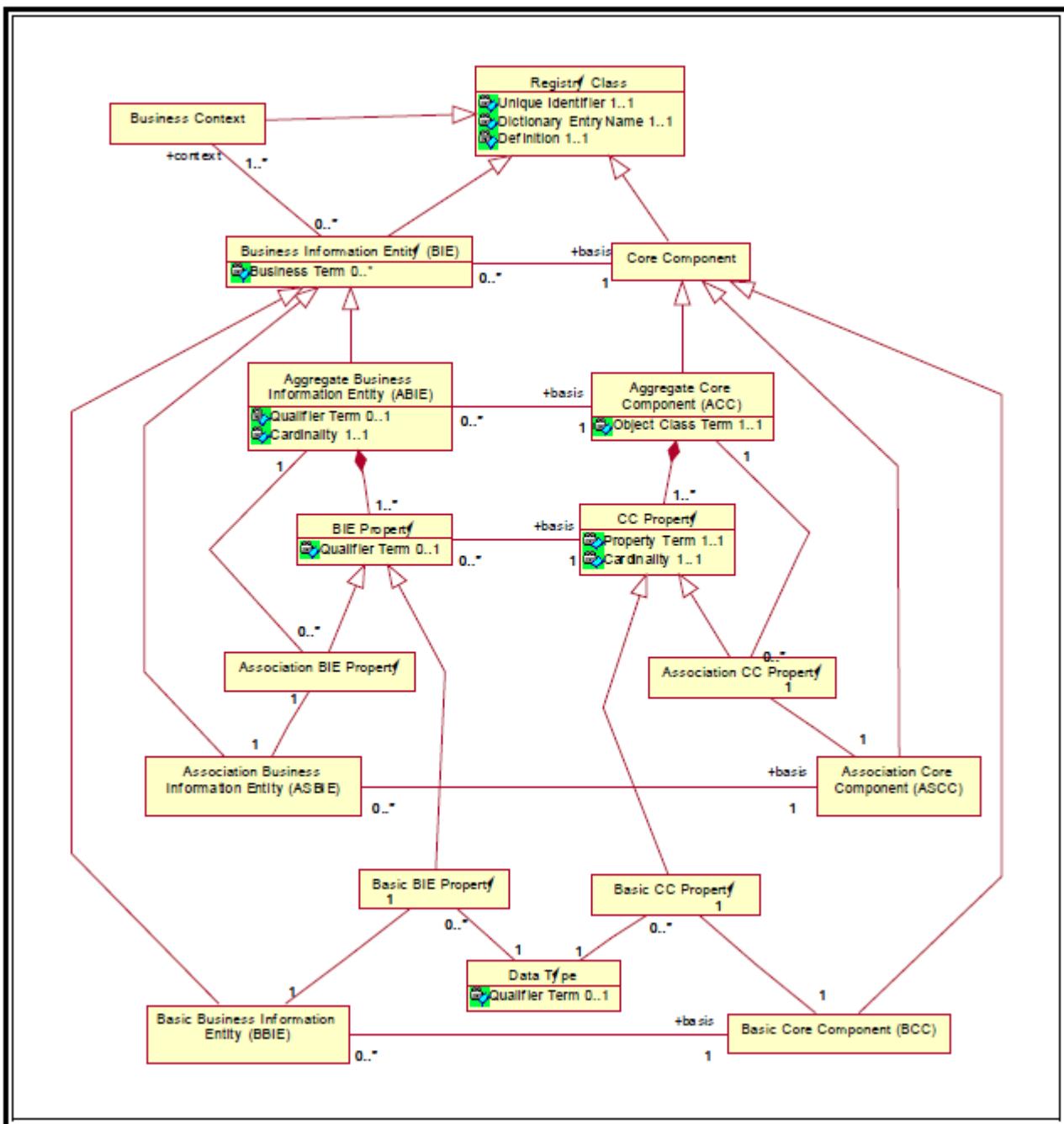
341 **5.2.2 Business Information Entities**

342 In the CCTS model, context neutral core components are instantiated as context specific
 343 components for business information payload and model harmonization. The context specific
 344 components are defined as Business Information Entities. (See CCTS Section 6.2 for a
 345 detailed discussion of the UN/CEFACT context mechanism.)³ Context specific CCTS Business

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

346 Information Entities are defined in CCTS as “A piece of business data or a group of pieces of
347 business data with a unique business semantic definition.”⁴

348 Figure 5-2 illustrates the various pieces of the overall `ccts:BusinessInformationEntity`
349 metamodel and their relationship with the `ccts:CoreComponents` metamodel.
350



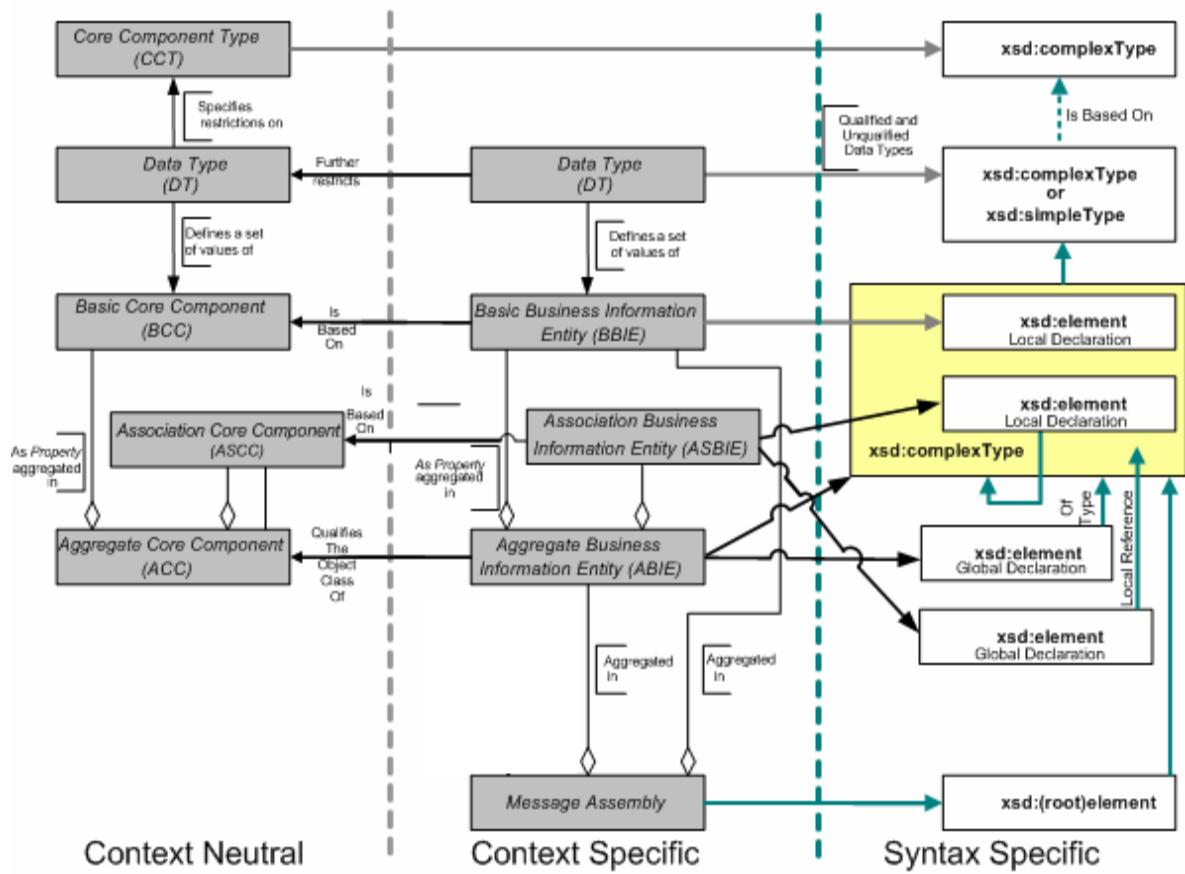
351
352 **Figure 5-2 Context Specific Business Information Entity Metamodel**

353 **5.2.3 The XML Constructs**

354 UN/CEFACT XML design rules are closely coupled with CCTS. UN/CEFACT XSD Schema will be
355 developed from fully conformant Business Information Entities that are based on fully conformant
356 Core Components. Figure 5-3 shows the relationship between CC's, BIE's and XSD artefacts. The
357 grey boxes reflect CCTS constructs (Core Component Types, Data Types, Core Components, and

⁴ Core Components Technical Specification, Part 8 of the ebXML Technical Framework Version 2.01, UN/CEFACT, 15 November 2003

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



360

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

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

377 [Note]

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

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

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

392 [Note]

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

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

402 5.3 Naming and Modelling Constraints

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

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

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

420 Example 5-1: Fully Qualified XPath

421 /CrossIndustryInvoice/CIExchangedDocumentContext/SpecifiedTransaction/Identifier
422 /Tender/ProcuringOrganization/Name/Text

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

- 427 [R6] Element, attribute and type names MUST be composed of words in the English
428 language, using the primary English spellings provided in the Oxford English
429 Dictionary.

430 Following the *ebXML Architecture Specification* and commonly used best practice, Lower Camel
431 Case (LCC) is used for naming attributes and Upper Camel Case (UCC) is used for naming
432 elements and types. Lower Camel Case capitalizes the first character of each word except the first
433 word and compounds the name. Upper Camel Case capitalizes the first character of each word and
434 compounds the name.

- 435 [R7] Lower camel case (LCC) MUST be used for naming attributes

- 437 **Example 5-2: Attribute**
- 438

```
<xsd:attribute name="unitCode" .../>
```
- 439
- 440 [R8] Upper camel case (UCC) MUST be used for naming elements and types.
- 441 **Example 5-3: Element**
- 442

```
<xsd:element name="LastReportedSubmissionDateTime" ...>
```
- 443 **Example 5-4: Type**
- 444

```
<xsd:complexType name="DocumentCodeType">
```
- 445
- 446 [R9] Element, attribute and type names MUST be in singular form unless the concept itself is plural.
- 447
- 448 **Example 5-5: Singular and Plural Concept Form**
- 449 **Allowed - Singular:**
- 450

```
<xsd:element name="GoodsCharacteristic" ...>
```
- 451 **Not Allowed - Plural:**
- 452

```
<xsd:element name="ItemsQuantity" ...>
```
- 453
- 454 [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.
- 455
- 456
- 457 **Example 5-6: Non-Letter Characters**
- 458 **Not Allowed**
- 459

```
<xsd:element name="LanguageCode8" ...>
```
- 460 The CCTS allows for the use of periods, spaces and other separators in the dictionary entry name.
- 461 XML best practice is to not include these in an XML tag name. Additionally, XML 1.0 specifically prohibits the use of certain reserved characters in XML tag names.
- 462
- 463 [R11] This rule has been combined with [R10].
- 464 **Example 5-7: Spaces in Name**
- 465 **Not Allowed**
- 466

```
<xsd:element name="Customized_ Language. Code:8" ...>
```
- 467
- 468 [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.
- 469
- 470
- 471 [R13] The acronyms and abbreviations listed in Appendix C MUST always be used.
- 472
- 473
- 474 [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.
- 475
- 476 [R15] Acronyms MUST appear in all upper case for all element declarations and type definitions.
- 477 **Example 5-8: Acronyms and Abbreviations**
- 478 **Allowed – ID is an approved abbreviation**
- 479

```
<xsd:attribute name="currencyID"
```
- 480 **Not Allowed – Cd is not an approved abbreviation, if it was an approved abbreviation it must appear in all upper case**
- 481
- 482

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

483 5.3.1 Element Naming Conventions

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

489 5.4 Reusability Scheme (Informative)

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

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

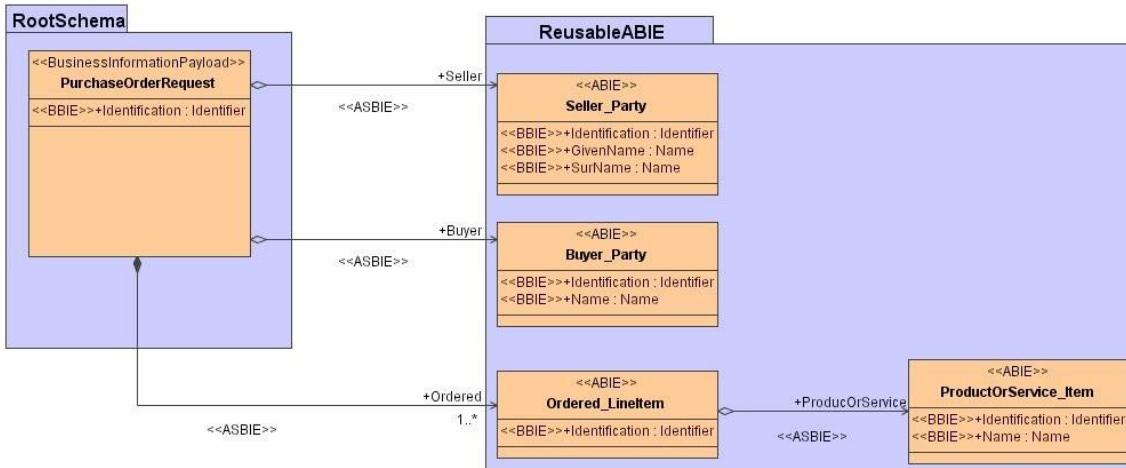
506 A purely type based approach does, however, limit the ability to reuse elements, especially in
507 technologies such as Web Services Description Language (WSDL). UN/CEFACT has thus decided
508 to implement what is known as a “hybrid approach” as this provides benefits over a purely type
509 based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

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

- 512 • All classes (PurchaseOrderRequest, Seller_Party, Buyer_Party, Ordered_LineItem and
513 ProductOrService_Item in figure 5-4) are declared as a **xsd:complexType**.
- 514 • All attributes of a class are declared as a local **xsd:element** within a **xsd:complexType**.
- 515 • Composition associations (e.g. PurchaseOrderRequest. Ordered. Ordered_LineItem in figure
516 5-4) are locally declared as a **xsd:element** within a **xsd:complexType**. A composition
517 ASBIE is defined as a specialized type of ASBIE that represents a composition
518 relationship between the associating ABIE and the associated ABIE.
- 519 • Associations that are not defined as composites (e.g. PurchaseOrderRequest.Buyer.
520 Buyer_Party, PurchaseOrderRequest. Seller. SellerParty in figure 5-4) are globally
521 declared as a **xsd:element**.

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

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



525

526

527

528

Figure 5-4 UML model example**Example 5-9: xsd declarations representing Figure 5-4**

```

529 <xsd:element name="PurchaseOrderRequest" type="rsm:PurchaseOrderRequestType"/>
530 <xsd:element name="BuyerParty" type="ram:BuyerPartyType"/>
531 <xsd:element name="OrderedLineItem" type="ram:OrderedLineItemType"/>
532 <xsd:element name="ProductOrServiceItem" type="ram:ProductOrServiceItemType"/>
533 <xsd:element name="SellerParty" type="ram:SellerPartyType"/>
534 <xsd:complexType name="PurchaseOrderRequestType">
535     <xsd:sequence>
536         <xsd:element name="ID" type="udt:IDType"/>
537         <xsd:element ref="ram:SellerParty"/>
538         <xsd:element ref="ram:BuyerParty"/>
539         <xsd:element name="OrderedLineItem"
540             type="ram:OrderedLineItemType" maxOccurs="unbounded"/>
541     </xsd:sequence>
542 </xsd:complexType>
543 <xsd:complexType name="BuyerPartyType">
544     <xsd:sequence>
545         <xsd:element name="ID" type="udt:IDType"/>
546         <xsd:element name="Name" type="udt:NameType"/>
547     </xsd:sequence>
548 </xsd:complexType>
549 <xsd:complexType name="OrderedLineItemType">
550     <xsd:sequence>
551         <xsd:element name="ID" type="udt:IDType"/>
552         <xsd:element name="ProductOrServiceItem" type="ram:ProductOrServiceItemType"/>
553     </xsd:sequence>
554 </xsd:complexType>
555 <xsd:complexType name="ProductOrServiceItemType">
556     <xsd:sequence>
557         <xsd:element name="ID" type="udt:IDType"/>
558         <xsd:element name="Name" type="udt:NameType"/>
559     </xsd:sequence>
560 </xsd:complexType>
561 <xsd:complexType name="SellerPartyType">
562     <xsd:sequence>
563         <xsd:element name="ID" type="udt:IDType"/>
564         <xsd:element name="GivenName" type="udt:NameType"/>
565         <xsd:element name="Surname" type="udt:NameType"/>
566     </xsd:sequence>
567 </xsd:complexType>
568
569
570
571
572

```

573

5.5 Modularity Model

Modularity in schema design promotes reuse and provides significant management capabilities. Modules can be either unique in their functionality, or represent splitting of larger schema files for

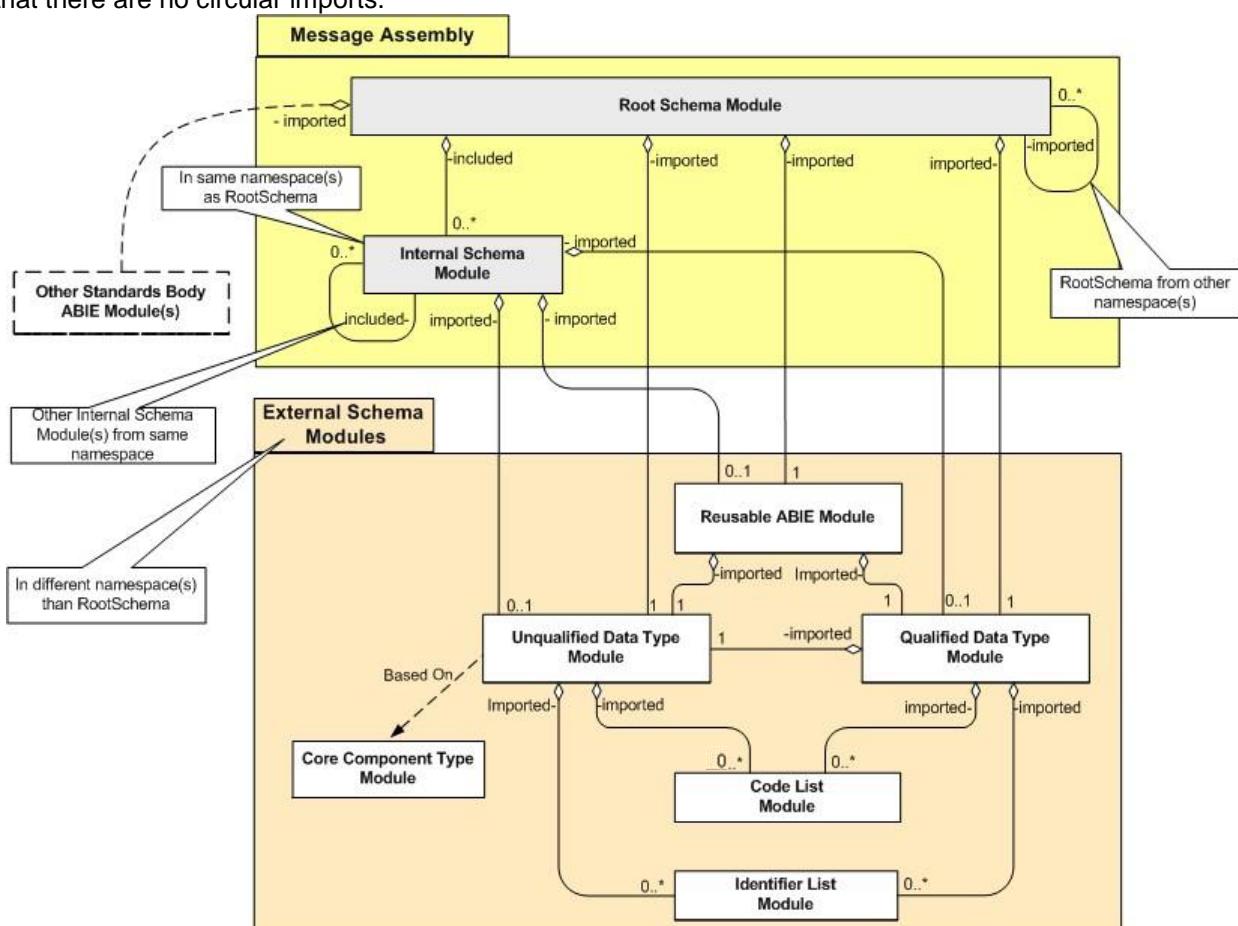
576 performance or manageability enhancement. A modularity model provides an efficient and effective
577 mechanism for importing and including components as needed rather than dealing with complex,
578 multi-focused schema.

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

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

594 The reusable ABIE schema module always imports the unqualified data type and qualified data type
595 schema modules. The unqualified data type schema imports necessary code list schema modules
596 and may import identifier list schema modules. The qualified data type schema modules always
597 import the unqualified data type schema module as well as necessary code list and identifier list
598 schema modules.

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



602

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

604 To ensure consistency, and for standardization of namespace tokens as addressed elsewhere in
605 this specification, all schema modules identified above are referred to by their formal name or
606 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 |

607

608 [R16] The schema module file name for modules other than code lists or identifier lists MUST
609 of the form <SchemaModuleName>_<Version>.xsd, with periods, spaces, or other
610 separators and the words Schema Module removed.

611 [R17] The schema module file name for code lists and identifier lists, MUST be of the form
612 <AgencyName>_<ListName>_<Version>.xsd, with periods, spaces, or other
613 separators removed.

614 [R18] In representing versioning schemes in file names, only the major version should be
615 included.

5.5.1 Root Schema

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

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

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

625 [R20] Each UN/CEFACT root schema module MUST be named
626 <BusinessInformationPayload> Schema Module.

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

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

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

638 [R22] UN/CEFACT XSD schema modules MUST either be treated as external schema
639 modules, or as internal schema modules of the root schema.

641 5.5.2 Internal Schema

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

648 Internal schema modules will reside in the same namespace as their parent root schema. Since the
649 internal schema reside in the same namespace as the root, the root schema uses `xsd:include` to
650 incorporate these internal modules. The UN/CEFACT XSD schema modularity approach ensures
651 that logical associations exist between root and internal schema modules and that individual schema
652 modules can be reused to the maximum extent possible.

-
- 653 [R23] All UN/CEFACT internal schema modules MUST be in the same namespace as their
654 corresponding `rsm:RootSchema`.
-

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

-
- 658 [R24] Each UN/CEFACT internal schema module MUST be named
659 `<ParentRootSchemaModuleName><InternalSchemaModuleFunction>` Schema
660 Module
-

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

662 TravelReservationRequestFlightInformation
663 Where:
664 TravelReservationRequest represents the parent root schema module name
665 FlightInformation represents the internal schema module function

666 5.5.3 External Schema

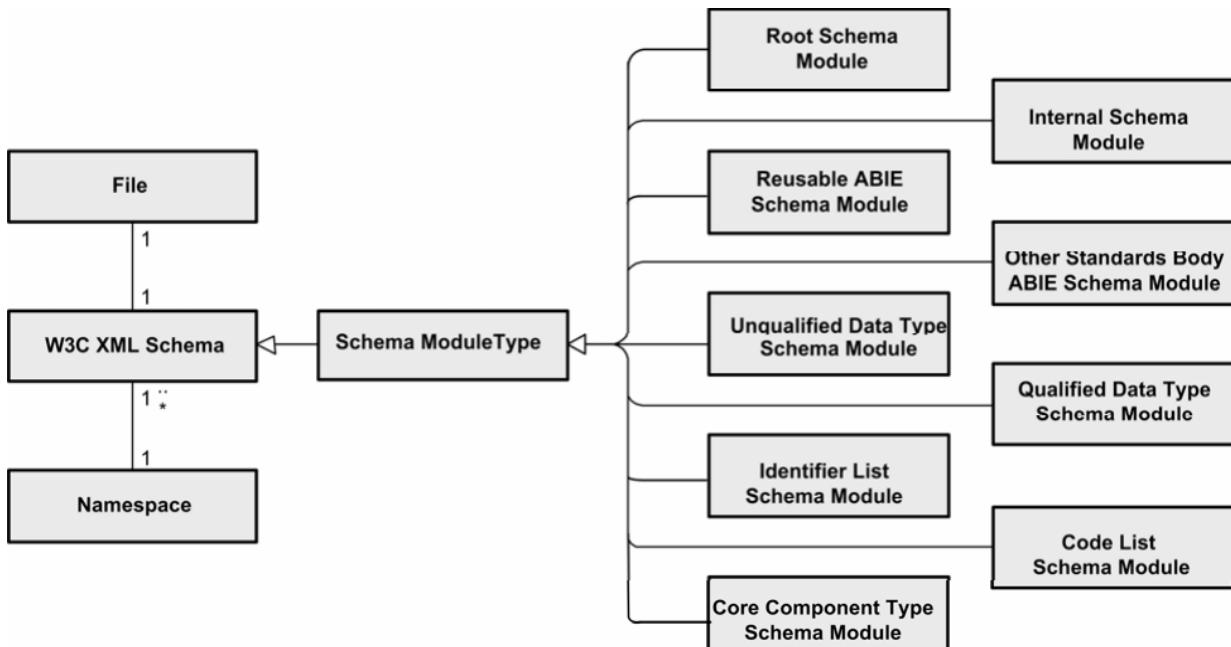
667 To adhere to the principles and rules contained in Section 7, schema modules will be created for
668 reusable components. These schema modules are referred to as external schema modules because
669 they reside in a different namespace from the root schema. Root schema may import one or more
670 of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

- 672
 - 673 o Unqualified Data Type
 - 674 o Qualified Data Type
 - 675 o Reusable ABIE
 - 676 o Code List
 - 677 o Identifier List
 - 677 o Other Standards Body ABIE module

678 [Note]

679 The terms “unqualified data type” and “qualified data type” refer to the ISO 11179
680 concept of qualifiers for name constructs, not to the XML namespace concept of
681 qualified and unqualified

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



683
684

Figure 5-6 UN/CEFACT XSD Schema Modules

685

5.5.3.1 Core Component Type Schema Module

686
687
688
689

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

690

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

691
692

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

693
694

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

695

5.5.3.2 Unqualified Data Type Schema Module

696
697
698
699
700

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

701
702

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

703
704

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

705

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

706
707

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

708
709

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

710

5.5.3.3 Qualified Data Type Schema Module

711
712
713

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

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

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

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

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

721 **5.5.3.4 Reusable Aggregate Business Information Entity Schema Module**

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

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

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

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

735 **5.5.3.5 Code List Schema Modules**

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

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

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

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

746 Where:

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

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

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

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

752 **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
```

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

754 Planning Level Code - Code List Schema Module

755 **5.5.3.6 Identifier List Schema Modules**

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

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

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

-
- 766 [R36] The name of each `ids:IdentifierList` schema module MUST be of the form:
767 `<Identifier Scheme Agency Identifier|Identifier Scheme Agency`
768 `Name><Identifier Scheme Identifier|Identifier Scheme Name>` -
769 `Identifier List Schema Module`

770 Where:

771 Identifier Scheme Agency Identifier = identification of the agency that maintains the
772 identifier list

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

774 Identifier Scheme Identifier = identification of the identifier list

775 Identification Scheme Name = Name as assigned by the agency that maintains the
776 identifier list

777 **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
```

778 **5.5.3.7 Other Standards Body Aggregate Business Information Entity Schema
779 Modules**

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

-
- 784 [R37] Imported schema modules MUST be fully conformant with the UN/CEFACT XML
785 *Naming and Design Rules Technical Specification* and the UN/CEFACT Core
786 *Components Technical Specification*.
-

787 **5.6 Namespace Scheme**

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

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

⁵ World Wide Web Consortium, *Namespaces in XML*, 14 January 1999

5.6.1 Namespace Scheme

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

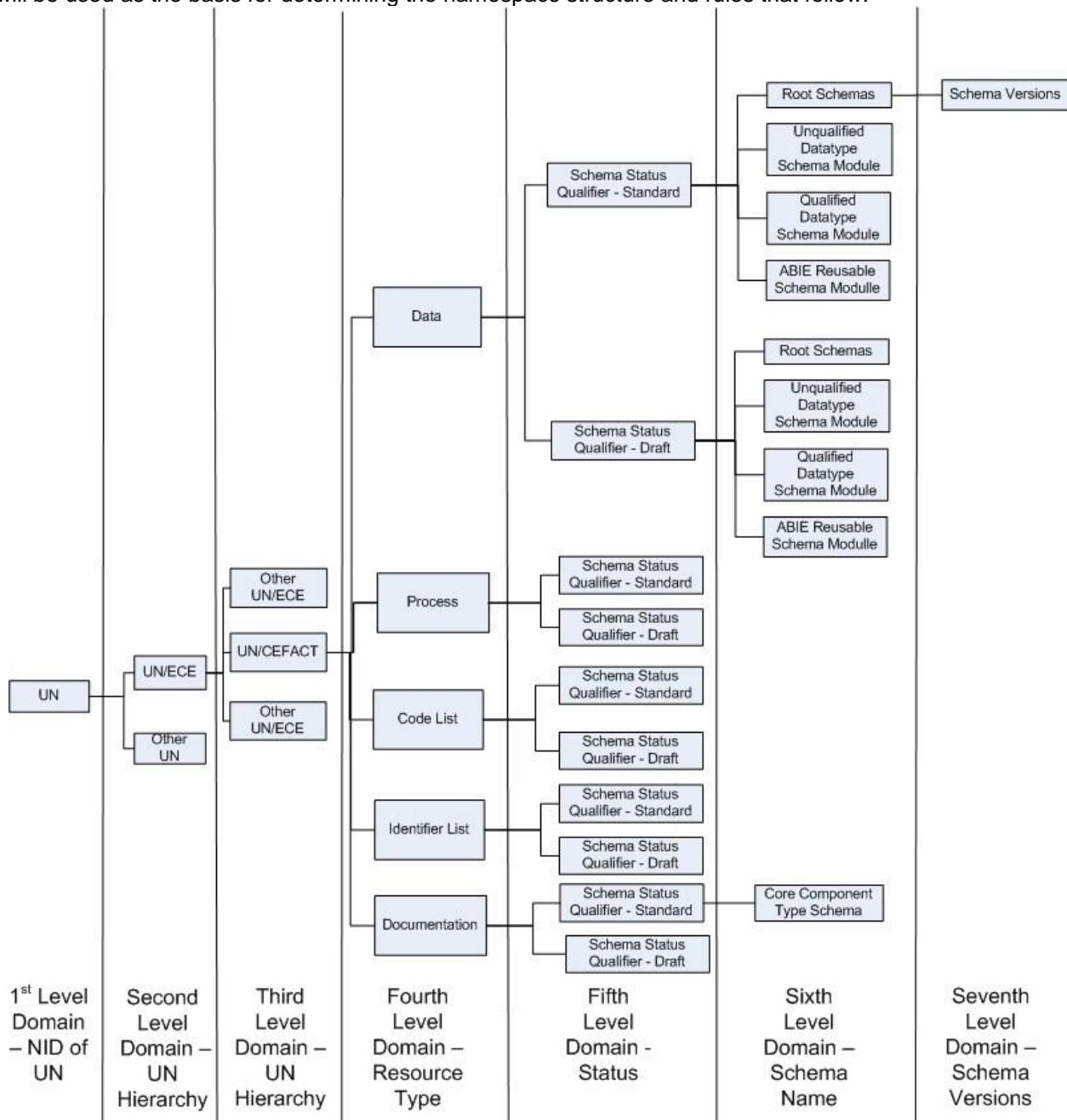


Figure 5-7: UN/CEFACT Namespace Scheme

5.6.2 Declaring Namespace

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

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

5.6.3 Namespace Persistence

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

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

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

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

5.6.4 Namespace Uniform Resource Identifiers

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

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

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

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

where :

<NID> = the Namespace Identifier
<NSS> = the Namespace Specific String.
The leading "urn:" sequence is case-insensitive.

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

Where:

- Namespace Identifier (NID) = un
- Namespace Specific String = `unece:uncefact:<schematype>:<status>:<name>:<version>` with unece and uncefact as fixed value second and third level domains within the NID of un
- schematype = a token identifying the type of schema module:
`data|process|codelist|identifierlist|documentation`
- status = the status of the schema as: `draft|standard`
- name = the name of the schema module (using upper camel case) with periods, spaces, or other separators and the words 'schema module' removed.
- version = The major version number. Sequentially assigned, first release starting with the number 1.

- 866 [R42] The names for namespaces MUST have the following structure while the schema is at
867 draft status:
868 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>`
869 Where:
870 schematype = a token identifying the type of schema module:
871 `data|process|codelist|identifierlist|documentation`
872 status = a token identifying the standards status of the schema module:
873 `draft|standard`
874 name = the name of the schema module (using upper camel case) with periods, spaces,
875 or other separators and the words 'schema module' removed.
876 major = the major version number. Sequentially assigned, first release starting with the
877 number 1.
878
- 879 [R43] This rule was combined with [R42].

880 **Example 5-14: Namespace Name at Draft Status**

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

882 **Example 5-15: Namespace Name at Specification Status**

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

884 **5.6.5 Namespace Constraint**

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

- 889 [R44] UN/CEFACT namespace values will only be assigned to UN/CEFACT developed
890 objects.

891 **5.6.6 UN/CEFACT XSD Namespace Schema Tokens**

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

896 **5.7 Schema Location**

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

- 904 [R45] The general structure for schema location MUST be:
905 `./<schematype>/<status>/<name>_<major>.<minor>[p <revision>].xsd`
906 Where:
907 schematype = a token identifying the type of schema module:
908 `data|process|codelist|identifierlist|documentation`
909 status = the status of the schema as: `draft|standard`
910 name = the name of the schema module (using upper camel case) with periods,
911 spaces, or other separators and the words 'schema module' removed.
912 major = the major version number, sequentially assigned, first release starting with the
913 number 1.
914 minor = the minor version number within a major release, sequentially assigned, first
915 release starting with the number 0.

916 revision = sequentially assigned alphanumeric character for each revision of a minor
917 release. Only applicable where status = draft.

918 [R46] Each **xsd:schemaLocation** attribute declaration MUST contain a resolvable URL, and
919 in the case of an absolute path, a persistent URL.

920 [R47] This rule has been removed.

921 5.8 Versioning

922 The versioning scheme for UN/CEFACT XSD schema modules is composed of a major version
923 number and where appropriate, a minor version number. Major version numbers are reflected in the
924 namespace declaration while minor version numbers are only reflected in the schema location.
925 Major and minor version numbers are also declared in the version attribute in the **xsd:schema**
926 element.

927 [R48] The **xsd:schema** version attribute MUST always be declared.

928 [R49] The **xsd:schema** version attribute MUST use the following template:
929 **<xsd:schema ... version=".">**

930 [R50] Every schema version namespace declaration MUST have the URI of:
931 **urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>**

932 5.8.1 Major Versions

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

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

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

943 **urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>** Where:

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

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

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

950 5.8.2 Minor Versions

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

- 957 ○ Adding values to enumerations
- 958 ○ Optional extensions
- 959 ○ Add optional elements

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

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

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

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

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

978 Semantic compatibility across minor versions is essential.

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

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

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

988

6 General XML Schema Language Conventions

989

6.1 Schema Construct

990

[R56] The `xsd:elementFormDefault` attribute MUST be declared and its value set to `qualified`.

992

[R57] The `xsd:attributeFormDefault` attribute MUST be declared and its value set to `unqualified`.

994

[R58] The `xsd` prefix MUST be used in all cases when referring to <http://www.w3.org/2001/XMLSchema> as follows:
`xmlns:xsd=http://www.w3.org/2001/XMLSchema`.

997

Example 6-1: Element and Attribute Form Default

998

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

999

1000

1001

6.1.1 Constraints on Schema Construction

1002

[R59] `xsd:appInfo` MUST NOT be used.

1004

[R60] `xsd:notation` MUST NOT be used.

1005

[R61] `xsd:wildcard` MUST NOT be used.

1006

[R62] The `xsd:any` element MUST NOT be used.

1007

[R63] The `xsd:any` attribute MUST NOT be used.

1008

[R64] Mixed content MUST NOT be used (excluding documentation).

1009

[R65] `xsd:substitutionGroup` MUST NOT be used.

1010

[R66] `xsd:ID/xsd:IDREF` MUST NOT be used.

1011

[R67] `xsd:key/xsd:keyref` MUST be used for information association.

1012

[R68] The absence of a construct or data MUST NOT carry meaning.

1013

6.2 Attribute and Element Declarations

1014

6.2.1 Attributes

1015

6.2.1.1 Usage of Attributes

1016

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

1019

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

1021

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

1023

[R70] A `xsd:attribute` that represents a supplementary component with variable information MUST be based on the appropriate XSD built-in data type.

1025

[R71] A `xsd:attribute` that represents a supplementary component which represents codes MUST be based on the `xsd:simpleType` of the appropriate code list.

1027 [R72] A **xsd:attribute** that represents a supplementary component which represents
1028 identifiers MUST be based on the **xsd:simpleType** of the appropriate identifier
1029 scheme.

1030 **6.2.1.2 Constraints on Attribute Declarations**

1031 In general, the absence of an element in an XML schema does not have any particular meaning - it
1032 may indicate that the information is unknown, or not applicable, or the element may be absent for
1033 some other reason. The XML schema specification does however provide a feature, the
1034 **xsd:nillable** attribute, whereby an element may be transferred with no content, but still use its
1035 attributes and thus carry semantic meaning. In order to respect the principles of the CCTS and to
1036 retain semantic clarity the nillability feature of XSD will not be used.

1037 [R73] The **xsd:nillable** attribute MUST NOT be used.

1038 **6.2.2 Elements**

1039 **6.2.2.1 Usage of Elements**

1040 Elements are declared for the document level business information payload, ABIEs, BBIEs, and
1041 ASBIEs.

1042 **6.2.2.2 Element Declaration**

1043 [R74] Empty elements MUST NOT be used.

1044 [R75] Every BBIE leaf element declaration MUST be of the **udt:UnqualifiedDataType** or
1045 **qdt:QualifiedDataType** that represents the source basic business information
1046 entity (BBIE) data type.

1047 Example 6-2: Element Declaration

```
1048 <xsd:complexType name="AcknowledgementType">
1049   <xsd:annotation>
1050     ... see annotation ...
1051   </xsd:annotation>
1052   <xsd:sequence>
1053     <xsd:element name="AcknowledgementDocument"
1054       type="ram:AcknowledgementDocumentType" minOccurs="0">
1055       <xsd:annotation>
1056         action>
1057         ... see annotation ...
1058       </xsd:annotation>
1059     </xsd:element>
1060     <xsd:element name="ProjectParty" type="ram:ProjectPartyType">
1061       <xsd:annotation>
1062         ... see annotation ...
1063       </xsd:annotation>
1064     </xsd:element>
1065   </xsd:sequence>
1066 </xsd:complexType>
```

1069 **6.2.2.3 Constraints on Element Declarations**

1070 [R76] The **xsd:all** element MUST NOT be used.

1071 **6.3 Type Declarations**

1072 **6.3.1 Usage of Types**

1073 [R77] All type definitions MUST be named.

1075
1076
1077
1078
1079
1080
1081
1082
1083

Example 6-3: Type Definition Name

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

1084 Data types are intended to be reused to the maximum extent possible. If an existing data type has
1085 the same semantic meaning and structure (facet restrictions) as the intended data type, then the
1086 existing data type should be used rather than creating a semantically equivalent duplicate data type.

1087 [R78] Data type definitions with the same semantic meaning MUST NOT have an identical set
1088 of facet restrictions.

6.3.2 Simple Type Definitions

1090 **xsd:simpleTypes** must always be used where they satisfy the user's business requirements.
1091 Where these business requirements cannot be satisfied, user defined complex type definitions will
1092 be used.

Example 6-4: Simple Types in Unqualified Data Type Schema Module

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

Example 6-5: Simple Types in Code Lists Module

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

6.3.3 Complex Type Definitions

1114 User defined complex types may be used when XSD built-in data types do not satisfy the business
1115 requirements or when an aggregate business information entity (ABIE) must be defined.

Example 6-6: Complex Type of Object Class "ProjectContactType"

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

6.4 User of XSD Extension and Restriction

1126 The general philosophy is that all UN/CEFACT XSD schema constructs will follow the model defined
1127 in Figure 5.1. These schema constructs are based on the concept that the underlying semantic
1128 structures of the core components and business information entities are normative forms of
1129 standards that developers are not allowed to alter without coordination of appropriate UN/CEFACT
1130 Domains. Accordingly, as business requirements dictate, new schema constructs will be created
1131 and new types defined and elements declared as appropriate. The concept of derivation through the

1132 use of **xsd:extension** and **xsd:restriction** will only be used in limited circumstances as
1133 described below.

1134 6.4.1 Extension

1135 [R79] **xsd:extension** MUST only be used in the **cct:CoreComponentType** schema
1136 module and the **udt:UnqualifiedDataType** schema module. When used it MUST
1137 only be used for declaring **xsd:attributes** to accommodate relevant supplementary
1138 components.

1139 6.4.2 Restriction

1140 The CCTS specification employs the concept of semantic restriction in creating specific instantiations
1141 of core components. Accordingly, **xsd:restriction** will be used as appropriate to define types
1142 that are derived from the existing types. Where used, the derived types must always be renamed.
1143 Simple and complex type restrictions may be used. **xsd:restriction** can be used for facet
1144 restriction and/or attribute restriction.

1145 [R80] When **xsd:restriction** is applied to a **xsd:simpleType** or **xsd:complexType**
1146 that represents a data type the derived construct MUST use a different name.

1147 Example 6-7: Restriction of Simple Type

```
1148 <xsd:simpleType name="TaxAmountType">
1149   <xsd:annotation>
1150     ...
1151     see
1152     annotation ...
1153   </xsd:annotation>
1154   <xsd:restriction base="udt:AmountType">
1155     <xsd:totalDigits value="10"/>
1156     <xsd:fractionDigits value="3"/>
1157   </xsd:restriction>
1158 </xsd:simpleType>
```

1158 6.5 Annotation

1159 All UN/CEFACT XSD schema constructs will use **xsd:annotation** to provide the documentation
1160 specified in Section 7 of CCTS.

1161 [R81] Each UN/CEFACT defined or declared construct MUST use the **xsd:annotation**
1162 element for required CCTS documentation.

1163 [Note]

1164 In order to conform to this specification, this rule also applies to any construct imported
1165 from other standards bodies.

1166 6.5.1 Documentation

1167 The annotation documentation will be used to convey all metadata as specified in the CCTS, i.e., to
1168 convey the semantic content carried in the XML construct. Therefore, all elements specified for the
1169 documentation are defined in the Core Component Technical Specification namespace. The
1170 current version of this namespace is:

1171 **urn:un:unece:uncefact:documentation:standard:CoreComponentsTechnicalSpeci**
1172 **fication:2.**

1173 Thus, all schema modules must contain the following namespace declaration:

1174 **ccts="urn:un:unece:uncefact:documentation:standard:CoreComponentsTechnic**
1175 **alSpecification:2."**

1176 and all documentation elements must be prefixed with 'ccts'.

1177 The following annotations are required as defined in section 7 in type definitions and element
1178 declarations (the representation of each item in XML code is shown in parenthesis):

- 1179
 - **Unique Identifier:** The unique identifier assigned to the artefact in the library. (UniqueId)

- 1180 ○ **Acronym:** The abbreviation of the type of component.
- 1181 (Acronym)
- 1182 ■ **BBIE** – Basic Business Information Entity
- 1183 ■ **ABIE** – Aggregate Business Information Entity
- 1184 ■ **ASBIE** – Associated Business Information Entity
- 1185 ■ **CCT** – Core Component Type
- 1186 ■ **QDT** – Qualified Data Type
- 1187 ■ **UDT** – Unqualified Data Type
- 1188 ○ **Dictionary Entry Name:** The complete name (not the tag name) of the artefact in the library. (DictionaryEntryName)
- 1189 ○ **Name:** The name of the supplementary component or business information payload. (Name)
- 1190 ○ **Version:** The version of the artefact as assigned by the registry. (Version)
- 1191 ○ **Definition:** The semantic meaning of the artefact. (Definition)
- 1192 ○ **Cardinality:** An indication of whether the property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the object. (Cardinality)
- 1193 ○ **Object Class Term:** The Object Class represented by the artefact. (ObjectClassTerm)
- 1194 ○ **Object Class Qualifier Term:** A term(s) that qualifies the Object Class. (ObjectClassQualifierTerm)
- 1195 ○ **Property Term:** The Property Term represented by the artefact. (PropertyTerm)
- 1196 ○ **Property Qualifier Term:** A term(s) that qualifies the Property Term. (PropertyQualifierTerm)
- 1197 ○ **Associated Object Class Term:** The Associated Object Class Term represented by the artefact. (AssociatedObjectClassTerm)
- 1198 ○ **Associated Object Class Qualifier Term:** A term(s) that qualifies the Associated Object ClassTerm. (AssociatedObjectClassQualifierTerm)
- 1199 ○ **Association Type:** The association type of the Association Business Information Entity. (AssociationType)
- 1200 ○ **Primary Representation Term:** The Primary Representation Term represented by the artefact. (PrimaryRepresentationTerm)
- 1201 ○ **Data Type Qualifier Term:** A term(s) that qualifies the Data Type Term. (DataTypeQualifierTerm)
- 1202 ○ **Primitive Type:** The primitive data type as assigned to the artefact by CCTS. (PrimitiveType)
- 1203 ○ **Business Process Context Value:** A valid value describing the Business Process contexts for which this construct has been designed. Default is 'In All Contexts'. (BusinessProcessContextValue)
- 1204 ○ **Geopolitical/Region Context Value:** A valid value describing the Geopolitical/Region contexts for which this construct has been designed. Default is 'In All Contexts'. (GeopoliticalOrRegionContextValue)
- 1205 ○ **Official Constraints Context Value:** A valid value describing the Official Constraints contexts for which this construct has been designed. Default is 'None'. (OfficialConstraintContextValue)
- 1206 ○ **Product Context Value:** A valid value describing the Product contexts for which this construct has been designed. Default is 'In All Contexts'. (ProductContextValue)
- 1207 ○ **Industry Context Value:** A valid value describing the Industry contexts for which this construct has been designed. Default is 'In All Contexts'. (IndustryContextValue)
- 1208 ○ **Business Process Role Context Value:** A valid value describing the Role contexts for which this construct has been designed. Default is 'In All Contexts'. (BusinessProcessRoleContextValue)
- 1209 ○ **Supporting Role Context Value:** A valid value describing the Supporting Role contexts for which this construct has been designed. Default is 'In All Contexts'. (SupportingRoleContextValue)
- 1210 ○ **System Capabilities Context Value:** A valid value describing the Systems Capabilities contexts for which this construct has been designed. Default is 'In All Contexts'. (SystemCapabilitiesContextValue)

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

1236
1237 Appendix F specifies normative information on the specific annotation required for each of the
1238 artefacts.
1239

1240
1241 Note: The list above defines the minimum annotation documentation requirements.
1242 However, additional annotation documentation may be included when necessary.

1243 Example 6-8: Example of annotation

```

1244 <xsd:annotation>
1245   <xsd:documentation xml:lang="en">
1246     <ccts:UniqueID>UN01005559</ccts:UniqueID>
1247     <ccts:Acronym>BBIE</ccts:Acronym>
1248     <ccts:DictionaryEntryName>CI Note. Content. Code</ccts:DictionaryEntryName>
1249     <ccts:Version>1.0</ccts:Version>
1250     <ccts:Definition>The code specifying the content of this CI
1251     note.</ccts:Definition>
1252     <ccts:Cardinality>0..1</ccts:Cardinality>
1253     <ccts:ObjectClassTerm>Note</ccts:ObjectClassTerm>
1254     <ccts:ObjectClassQualifierTerm>CI</ccts:ObjectClassQualifierTerm>
1255     <ccts:PropertyTerm>Content</ccts:PropertyTerm>
1256     <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>
1257     <ccts:BusinessProcessContextValue>Cross Industry
1258       Trade</ccts:BusinessProcessContextValue>
1259     <ccts:GeopoliticalOrRegionContextValue>In All
1260       Contexts</ccts:GeopoliticalOrRegionContextValue>
1261     <ccts:OfficialConstraintContextValue>None</ccts:OfficialConstraintContextValue>
1262     <ccts:ProductContextValue>In All Contexts</ccts:ProductContextValue>
1263     <ccts:IndustryContextValue>In All Contexts</ccts:IndustryContextValue>
1264     <ccts:BusinessProcessRoleContextValue>In All
1265       Contexts</ccts:BusinessProcessRoleContextValue>
1266     <ccts:SupportingRoleContextValue>In All
1267       Contexts</ccts:SupportingRoleContextValue>
1268     <ccts:SystemCapabilitiesContextValue>In All
1269       Contexts</ccts:SystemCapabilitiesContextValue>
1270   </xsd:documentation>
1271 </xsd:annotation>
1272
1273

```

1274 Each UN/CEFACT construct containing a code should include documentation that will identify the
1275 code list(s) that must be minimally supported when the construct is used.

1276 The following table provides a summary view of the annotation data as defined in section 6.

| | rem:RootSchema | ABIE: xsd:complexType and xsdelement | BIE: xsd:element | ASBIE xsd:element | cct:CoreComponentType | Supplementary component | udt:UnqualifiedDataType | qdt:QualifiedDataType |
|--|----------------|--------------------------------------|------------------|-------------------|-----------------------|-------------------------|-------------------------|-----------------------|
| 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 | |

| | tem:RootSchema | ABIE: xsd:complexType and xsdelement | BBIE: xsd:element | ASBIE xsd:element | cct:CoreComponentType | Supplementary component | udt:UnqualifiedDataType | qdt:QualifiedDataType |
|-------------------------------------|----------------|---|-------------------|-------------------|-----------------------|-------------------------|-------------------------|-----------------------|
| 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 |

1277

1278 Key:

1279 M - mandatory

1280 O - optional

1281 R - repeating

1282 Note

1283 When a particular optional annotation element contains no value, it may be omitted from the
 1284 schema.

7 XML Schema Modules

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

7.1 Root Schema

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

7.1.1 Schema Construct

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

Example 7-1: Structure of RootSchema Module

```
<?xml version="1.0" encoding="UTF-8"?>
<!!-- ===== [MODULENAME] Schema Module ===== -->
<!!-- ===== Schema agency: UN/CEFACT ===== -->
<!!-- ===== Schema version: 2.0 ===== -->
<!!-- ===== 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] ===== -->
<!!-- ===== See type definition .... ===== -->
<xsd:complexType name="[TYPENAME]">
  <xsd:restriction base="xsd:token">
    ... see type definition ....
  </xsd:restriction>
</xsd:complexType>
</xsd:schema>
```

1341 7.1.2 Namespace Scheme

1342 All root schemas published by UN/CEFACT will be assigned a unique token by BPS to represent the
1343 namespace prefix. This token will be prefixed by 'rsm'.

1344 [R82] The root schema module MUST be represented by a unique token.

1345 Example 7-2: Namespace of Root Schema Module

```
1346 "xmlns:rsm="urn:un:unece:uncefact:data:draft:CrossIndustryInvoice:1"
```

1347 [Note]

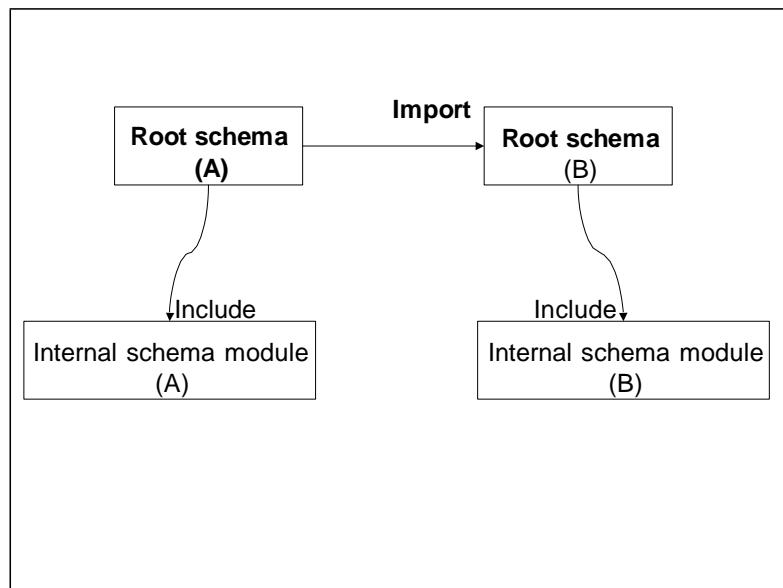
1348 Throughout this specification, the token 'rsm' is used for the unique root schema token.

1349 7.1.3 Imports and Includes

1350 [R83] The rsm:RootSchema MUST import the following schema modules:

- ram:ReusableABIE Schema Module
- udt:UnqualifiedDataType Schema Module
- qdt:QualifiedDataType Schema Module

1354 The root schema will include all internal schema modules that reside in its namespace. The root schema may
1355 import other external schema modules as necessary provided they conform to UN/CEFACT naming and
1356 design rules. One root schema (root schema A) may also make use of ABIEs defined as part of another root
1357 schema (root schema B) or that root schema's internal schema module. In other words, reuse type
1358 definitions and element declarations defined in another namespace. An example may be that the root schema
1359 for a Purchase Order Response message (root schema A) makes use of ABIEs defined as part of the schema
1360 definition for a Purchase Order Request message (root schema B). If that is the case then such type
1361 definitions and element declarations should be imported in to the root schema (root schema A). To achieve
1362 this only the root schema (root schema B) in the namespace containing the type definitions and element
1363 declarations needed should be imported as this in itself included the subordinate internal schema modules.



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.⁶

- 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>
```

⁶ 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 MUST have a structured set of annotations
1327 present in the following pattern:
- 1328 o UniqueID (mandatory): The identifier that references the business information payload
1329 instance in a unique and unambiguous way.
 - 1330 o Acronym (mandatory): The abbreviation of the type of component. In this case the value will
1331 always be RSM.
 - 1332 o Name (mandatory): The name of the business information payload.
 - 1333 o Version (mandatory): An indication of the evolution over time of a business information
1334 payload.
 - 1335 o Definition (mandatory): A brief description of the business information payload.
 - 1336 o BusinessProcessContextValue (mandatory, repetitive): The business process with which this
1337 business information is associated.
 - 1338 o GeopoliticalRegionContextValue (optional, repetitive): The geopolitical/region contexts for
1339 this business information payload.
 - 1340 o OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
1341 business information payload.
 - 1342 o ProductContextValue (optional, repetitive): The product context for this business information
1343 payload.
 - 1344 o IndustryContextValue (optional, repetitive): The industry context for this business information
1345 payload.
 - 1346 o BusinessProcessRoleContextValue (optional, repetitive): The role context for this business
1347 information payload.
 - 1348 o SupportingRoleContextValue (optional, repetitive): The supporting role context for this
1349 business information payload.
 - 1350 o SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
1351 this business information payload.

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
1363 corresponding **rsm:RootSchema**.

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.

7.3 Reusable Aggregate Business Information Entity Schema

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.

7.3.1 Schema Construct

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.

Example 7-5: Structure of Reusable ABIEs Schema Module

```
<?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>
```

7.3.2 Namespace Scheme

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

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

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

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

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

7.3.3 Imports and Includes

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

- **udt:UnqualifiedDataType** Schema Module
- **qdt:QualifiedDataType** Schema Module

Example 7-8: Import of required modules

```
<!---- ===== 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 <!-- ===== Import of Unqualified Data Type Schema Module ===== -->
1427 <!-- ===== -->
1428 <xsd:import
1429   namespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataType:1"
1430   schemaLocation="http://www.unece.org/uncefact/data/draft/UnqualifiedDataTypes_1.xsd"/>
1431

```

7.3.4 Type Declarations

- 1432 [R97] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named **xsd:complexType** MUST be defined.
- 1433 [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 For every complex type definition based on an ABIE object class, its XSD content model will be defined such
1439 that it reflects each property of the object class as an element declaration, with its cardinality and sequencing
1440 within the schema XSD content model determined by the details of the source aggregate business information
1441 entity (ABIE).

- 1442 [R99] Every aggregate business information entity (ABIE) **xsd:complexType** definition content model
1443 MUST use the **xsd:sequence** and/or **xsd:choice** elements to reflect each property (BBIE or
1444 ASBIE) of its class.
- 1445 [R100] Recursion of **xsd:sequence** and/or **xsd:choice** MUST NOT occur.

1446 No complex type may contain a sequence followed by another sequence or a choice followed by another
1447 choice. However, it is permissible to alternate sequence and choice as in example 7.9. Note that the choice
1448 construction will not be used in the base reusable ABIE UN/CEFACT schemas, as it cannot be directly
1449 modeled in CCTS. However, third party schemas that implement those restrictions would still be
1450 conformant.

Example 7-9: Sequence within an object class

```

1452 <xsd:complexType name="AcknowledgementDocumentType" >
1453   <xsd:annotation>
1454     ...see annotation...
1455   </xsd:annotation>
1456   <xsd:sequence>
1457     <xsd:element name="MultipleReferencesIndicator" type="udt:IndicatorType"
1458       minOccurs="0 ">
1459       <xsd:annotation>
1460         ...see annotation...
1461       </xsd:annotation>
1462     </xsd:element>
1463     <xsd:element name="ID" type="udt:IDType" minOccurs="0" >
1464       <xsd:annotation>
1465         ...see annotation...
1466       </xsd:annotation>
1467     </xsd:element>
1468     <xsd:element name="TypeCode" type="qdt:DocumentTypeCode" minOccurs="0"
1469       maxOccurs="unbounded">
1470       <xsd:annotation>
1471         ...see annotation...
1472       </xsd:annotation>
1473     </xsd:element>
1474     ...
1475   </xsd:sequence>
1476 </xsd:complexType>

```

Example 7-10: Choice

```

1477 <xsd:complexType name="LocationType">
1478   <xsd:annotation>
1479     ... see annotation ...
1480   </xsd:annotation>
1481   <xsd:choice>
1482

```

```

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

```

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

```

1503 <xsd:complexType name="PeriodType">
1504     ...
1505     <xsd:sequence>
1506         <xsd:element name="DurationDateTime"
1507             type="qdt:DurationDateTimeType" minOccurs="0"
1508             maxOccurs="unbounded">
1509             ...
1510         </xsd:element>
1511         ...
1512         <xsd:choice>
1513             <xsd:sequence>
1514                 <xsd:element name="StartTime" type="udt:TimeType"
1515                     minOccurs="0">
1516                     ...
1517                 </xsd:element>
1518                 <xsd:element name="EndTime" type="udt:TimeType"
1519                     minOccurs="0">
1520                     ...
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>

```

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

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>
```

7.3.5 Element Declarations and References

1567 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 **ccts: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 **ccts:AssociationType** is Composition, and element references to the globally
1576 declared elements for ASBIEs whose **ccts: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 **ccts:AssociationType** is Composition will always be declared locally.

1590 [R109] For every ASBIE whose **ccts: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 who's **ccts: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 **ccts:AssociationType** is not a composition, the globally
1599 declared element for the associated ABIE must be referenced using **xsd:ref**.

1600 **Example 7-13: Element declaration and reference within an ABIE type definition**

```
1601   <xsd:complexType name="PurchaseOrderRequestType">
1602     <xsd:sequence>
1603       <xsd:element name="ID" type="udt:IDType"/>
1604       <xsd:element name="SellerParty" type="ram:SellerPartyType"/>
1605       <xsd:element name="BuyerParty" type="ram:BuyerPartyType"/>
1606       <xsd:element ref="ram:OrderedLineItem" maxOccurs="unbounded"/>
1607     </xsd:sequence>
1608   </xsd:complexType>
```

1609 **7.3.6 Annotation**

- 1610 [R113] For every ABIE **xsd:complexType** and **xsd:element** definition a structured set of
1611 annotations MUST be present in the following pattern:
- o UniqueID (mandatory): The identifier that references an ABIE instance in a unique and
1613 unambiguous way.
 - o Acronym (mandatory): The abbreviation of the type of component. In this case the value will
1615 always be ABIE.
 - o DictionaryEntryName (mandatory): The official name of an ABIE.
 - o Version (mandatory): An indication of the evolution over time of an ABIE instance.
 - o Definition (mandatory): The semantic meaning of an ABIE.
 - o ObjectClassTerm (mandatory): The Object Class Term of the ABIE.
 - o ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the ABIE.
 - o BusinessProcessContextValue (optional, repetitive): The business process with which this
1622 ABIE is associated.
 - o GeopoliticalRegionContextValue (optional, repetitive): The geopolitical/region contexts for
1624 this ABIE.
 - o OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
1626 ABIE.
 - o ProductContextValue (optional, repetitive): The product context for this ABIE.
 - o IndustryContextValue (optional, repetitive): The industry context for this ABIE.
 - o BusinessProcessRoleContextValue (optional, repetitive): The role context for this ABIE.
 - o SupportingRoleContextValue (optional, repetitive): The supporting role context for this ABIE.
 - o SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
1632 this ABIE.
 - o UsageRule (optional, repetitive): A constraint that describes specific conditions that are
1634 applicable to the ABIE.
 - o BusinessTerm (optional, repetitive): A synonym term under which the ABIE is commonly
1636 known and used in the business.
 - o Example (optional, repetitive): Example of a possible value of an ABIE.

1638 [R114] This rule was combined with [R113].

1640 **Example 7-14: Annotation of an ABIE**

```
1641   <xsd:complexType name="AgriculturalPlotType" >
1642     <xsd:annotation>
1643       <xsd:documentation xml:lang="en">
1644         <ccts:UniqueID>UN01002651</ccts:UniqueID>
1645         <ccts:Acronym>ABIE</ccts:Acronym>
1646         <ccts:DictionaryEntryName>Agricultural_Plot_Details</ccts:DictionaryEntryName>
1647         <ccts:Version>1.0</ccts:Version>
1648         <ccts:Definition>A small piece of land used in agriculture.</ccts:Definition>
1649         <ccts:ObjectClassTerm>Plot</ccts:ObjectClassTerm>
1650         <ccts:ObjectClassQualifierTerm>Agricultural</ccts:ObjectClassQualifierTerm>
1651         <ccts:BusinessProcessContextValue>Crop Data Sheet</ccts:BusinessProcessContextValue>
1652         <ccts:GeopoliticalRegionalContextValue>Global</ccts:
1653           GeopoliticalRegionalContextValue >
1654           <ccts:OfficialConstraintContextValue>European, National, Local
1655             Regulations</ccts:OfficialConstraintContextValue>
1656             <ccts:ProductContextValue>Arable crop</ccts:ProductContextValue>
1657             <ccts:IndustryContextValue>Agricultural</ccts:IndustryContextValue>
1658             <ccts:BusinessProcessRoleContextValue>In All
1659               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>
```

- 1667 [R115] For every BBIE **xsd:element** declaration a structured set of annotations MUST be present in
 1668 the following pattern:
- o UniqueID (mandatory): The identifier that references a BBIE instance in a unique and
 1669 unambiguous way.
 - o Acronym (mandatory): The abbreviation of the type of component. In this case the value will
 1670 always be BBIE.
 - o DictionaryEntryName (mandatory): The official name of the BBIE.
 - o VersionID (mandatory): An indication of the evolution over time of a BBIE instance.
 - o Definition (mandatory): The semantic meaning of the BBIE.
 - o Cardinality (mandatory): Indication whether the BBIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.
 - o ObjectClassTerm (mandatory): The Object Class Term of the parent ABIE.
 - o ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the parent ABIE.
 - o PropertyTerm (mandatory): The Property Term of the BBIE.
 - o PropertyQualifierTerm (optional): Qualifies the Property Term of the BBIE.
 - o PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the BBIE.
 - o BusinessProcessContextValue (optional, repetitive): The business process with which this
 1671 BBIE is associated.
 - o GeopoliticalRegionContextValue (optional, repetitive): The geopolitical/region contexts for
 1672 this BBIE.
 - o OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
 1673 BBIE.
 - o ProductContextValue (optional, repetitive): The product context for this BBIE.
 - o IndustryContextValue (optional, repetitive): The industry context for this BBIE.
 - o BusinessProcessRoleContextValue (optional, repetitive): The role context for this BBIE.
 - o SupportingRoleContextValue (optional, repetitive): The supporting role context for this BBIE.
 - o SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
 1674 this BBIE.
 - o UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 1675 applicable to this BBIE.
 - o BusinessTerm (optional, repetitive): A synonym term under which the BBIE is commonly
 1676 known and used in the business.
 - o Example (optional, repetitive): Example of a possible value of a BBIE.

1700 Example 7-15: Annotation of a BBIE

```

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_Identifier</ccts:DictionaryEntryName>
1707       <ccts:Version>1.0</ccts:Version>
1708       <ccts:Definition>The unique identifier for this agricultural plot.</ccts:Definition>
1709       <ccts:Cardinality>1</ccts:Cardinality>
1710       <ccts:ObjectClassTerm>Plot</ccts:ObjectClassTerm>
1711     </xsd:documentation>
1712   </xsd:annotation>
1713 </xsd:element>
```

```

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>

```

- 1733 [R116] For every ASBIE **xsd:element** declaration a structured set of annotations MUST be present in
1734 the following pattern:
- 1735 o UniqueID (mandatory): The identifier that references an ASBIE instance in a unique and
1736 unambiguous way.
 - 1737 o Acronym (mandatory): The abbreviation of the type of component. In this case the value will
1738 always be ASBIE.
 - 1739 o DictionaryEntryName (mandatory): The official name of the ASBIE.
 - 1740 o Version (mandatory): An indication of the evolution over time of the ASBIE instance.
 - 1741 o Definition (mandatory): The semantic meaning of the ASBIE.
 - 1742 o Cardinality (mandatory): Indication whether the ASBIE Property represents a not-applicable,
1743 optional, mandatory and/or repetitive characteristic of the ABIE.
 - 1744 o ObjectClassTerm (mandatory): The Object Class Term of the associating ABIE.
 - 1745 o ObjectClassQualifierTerm (optional): A term that qualifies the Object Class Term of the
1746 associating ABIE.
 - 1747 o PropertyTerm (mandatory): The Property Term of the ASBIE.
 - 1748 o PropertyQualifierTerm (Optional): A term that qualifies the Property Term of the ASBIE.
 - 1749 o AssociatedObjectClassTerm (mandatory): The Object Class Term of the associated ABIE.
 - 1750 o AssociatedObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the
1751 associated ABIE.
 - 1752 o AssociationType (mandatory): The Association Type of the ASBIE.
 - 1753 o BusinessProcessContextValue (optional, repetitive): The business process with which this
1754 ASBIE is associated.
 - 1755 o GeopoliticalorRegionContextValue (optional, repetitive): The geopolitical/region contexts for
1756 this ASBIE.
 - 1757 o OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
1758 ASBIE.
 - 1759 o ProductContextValue (optional, repetitive): The product context for this ASBIE.
 - 1760 o IndustryContextValue (optional, repetitive): The industry context for this ASBIE.
 - 1761 o BusinessProcessRoleContextValue (optional, repetitive): The role context for this ASBIE.
 - 1762 o SupportingRoleContextValue (optional, repetitive): The supporting role context for this
1763 ASBIE.
 - 1764 o SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
1765 this ASBIE.
 - 1766 o UsageRule (optional, repetitive): A constraint that describes specific conditions that are
1767 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       <cccts:UniqueID>UN01002659</cccts:UniqueID>
1777       <cccts:Acronym>ASBIE</cccts:Acronym>
1778       <cccts:DictionaryEntryName>Agricultural_Plot. Included In. Agricultural_
1779       Country Sub-Division</cccts:DictionaryEntryName>
1780       <cccts:Version>1.0</cccts:Version>
1781       <cccts:Definition>An agricultural country sub-division in which this
1782       agricultural plot is included.</cccts:Definition>
1783       <cccts:Cardinality>0..n</cccts:Cardinality>
1784       <cccts:ObjectClassTerm>Plot</cccts:ObjectClassTerm>
1785       <cccts:ObjectClassQualifierTerm>Agricultural</cccts:ObjectClassQualifierTe
1786       rm>
1787       <cccts:AssociationType>composition</cccts:AssociationType>
1788       <cccts:PropertyTerm>Included In</cccts:PropertyTerm>
1789       <cccts:AssociatedObjectClassTerm>Country Sub-
1790       Division</cccts:AssociatedObjectClassTerm>
1791       <cccts:AssociatedObjectClassQualifierTerm>Agricultural</cccts:AssociatedOb
1792       jectClassQualifierTerm>
1793       <cccts:BusinessProcessContextValue>Crop Data
1794       Sheet</cccts:BusinessProcessContextValue>
1795       <cccts:GeopoliticalOrRegionContextValue>Global</cccts:GeopoliticalOrRegion
1796       ContextValue>
1797       <cccts:OfficialConstraintContextValue>European, National, Local
1798       Regulations</cccts:OfficialConstraintContextValue>
1799       <cccts:ProductContextValue>Arable crop</cccts:ProductContextValue>
1800       <cccts:IndustryContextValue>Agricultural</cccts:IndustryContextValue>
1801       <cccts:BusinessProcessRoleContextValue>In All
1802       Contexts</cccts:BusinessProcessRoleContextValue>
1803       <cccts:SupportingRoleContextValue>In All
1804       Contexts</cccts:SupportingRoleContextValue>
1805       <cccts:SystemCapabilitiesContextValue>In All
1806       Contexts</cccts: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
1828   ... see intellectual property disclaimer ...

```

```

1828 -->
1829 <xsd:schema targetNamespace=
1830   ... see namespace ... xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1831   elementFormDefault="qualified" attributeFormDefault="unqualified">
1832   <!-- =====-->
1833   <!-- ===== Type Definitions =====-->
1834   <!-- =====-->
1835   <!-- ===== CCT: AmountType =====-->
1836   <!-- =====-->
1837   ... see type definitions ...
1838 </xsd:schema>
```

7.4.3 Namespace Scheme

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

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

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

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

```

1844 <xsd:schema
1845   targetNamespace="urn:un:unece:uncefact:documentation:draft:CoreComponentType:2"
1846   xmlns:cct="urn:un:unece:uncefact:documentation:draft:CoreComponentType:2"
1847   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1848   elementFormDefault="qualified" attributeFormDefault="unqualified">
```

7.4.4 Imports and Includes

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

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

7.4.5 Type Definitions

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

[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.

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

[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.

[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.

Example 7-20: Type definition of a CCT

```

1868 <!-- =====-->
1869 <!-- ===== Type Definitions =====-->
1870 <!-- =====-->
1871 <!-- ===== AmountType =====-->
1872 <!-- =====-->
1873 <xsd:complexType name="AmountType">
1874   <xsd:annotation>
1875     ... see annotation ...
1876   </xsd:annotation>
1877   <xsd:simpleContent>
1878     <xsd:extension base="xsd:decimal">
1879       <xsd:attribute name="currencyID" type="xsd:token" use="optional">
1880       <xsd:annotation>
```

```

1881           ... see annotation ...
1882           </xsd:annotation>
1883           </xsd:attribute>
1884           ... see attribute declaration ...
1885           </xsd:extension>
1886           </xsd:simpleContent>
1887       </xsd:complexType>
```

1888 7.4.6 Attribute Declarations

1889 The current CCTS does not specify the components of the CCT supplementary component dictionary entry
 1890 name. However, in order to ensure a standard approach to declaring the supplementary components as
 1891 attributes, BPS has applied the naming concepts from ISO 11179, part 5. Specifically, BPS has defined the
 1892 dictionary entry name as it is stated in CCTS in terms of object class, property term, and representation term.
 1893 These components are identified in the annotation documentation for each supplementary component in the
 1894 CCT schema module.

-
- 1895 [R124] Each core component type supplementary component **xsd:attribute** name MUST be the
 1896 CCTS supplementary component dictionary entry name with the separators and spaces
 1897 removed.
 - 1898 [R125] If the object class of the supplementary component dictionary entry name contains the name of
 1899 the representation term of the parent CCT, the duplicated object class word or words MUST be
 1900 removed from the supplementary component **xsd:attribute** name.
 - 1901 [R126] If the object class of the supplementary component dictionary entry name contains the term
 1902 'identification', the term 'identification' MUST be removed from the supplementary component
 1903 **xsd:attribute** name.
 - 1904 [R127] If the representation term of the supplementary component dictionary entry name is 'text', the
 1905 representation term MUST be removed from the supplementary component **xsd:attribute**
 1906 name.
 - 1907 [R128] The attribute representing the supplementary component MUST be based on the appropriate XSD
 1908 built-in data type.
-

1909 Example 7-21: Supplementary component other than code or identifier

```

1910 <xsd:complexType name="BinaryObjectType">
1911 ...
1912     <xsd:simpleContent>
1913         <xsd:extension base="xsd:base64Binary">
1914             <xsd:attribute name="format" type="xsd:string" use="optional">
1915                 ...
1916             </xsd:attribute>
1917             ...
1918         </xsd:extension>
1919     </xsd:simpleContent>
1920 </xsd:complexType>
```

1921 7.4.7 Extension and Restriction

1922 The core component type schema module is a generic module based on the underlying core component
 1923 types. No restriction or extension is appropriate.

1924 7.4.8 Annotation

-
- 1925 [R129] For every core component type **xsd:complexType** definition a structured set of annotations
 1926 MUST be present in the following pattern:
 - 1927 ○ UniqueID (mandatory): The identifier that references the Core Component Type instance in a
 1928 unique and unambiguous way.
 - 1929 ○ Acronym (mandatory): The abbreviation of the type of component. . In this case the value will
 1930 always be CCT.
 - 1931 ○ DictionaryEntryName (mandatory): The official name of a Core Component Type.
 - 1932 ○ Version (mandatory): An indication of the evolution over time of a Core Component Type
 1933 instance.
-

- Definition (mandatory): The semantic meaning of a Core Component Type.
 - PrimaryRepresentationTerm (mandatory): The primary representation term of the Core Component Type.
 - PrimitiveType (mandatory): The primitive data type of the Core Component Type.
 - UsageRule (optional, repetitive): A constraint that describes specific conditions that are 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 MUST be present in the following pattern:

- UniqueID (optional): The identifier that references the Supplementary Component instance in a unique and unambiguous way.
 - Acronym (mandatory): The abbreviation of the type of Supplementary Component. In this case the value will always be SC.
 - DictionaryEntryName (mandatory): The official name of the Supplementary Component.
 - Definition (mandatory): The semantic meaning of the Supplementary Component.
 - Cardinality (mandatory): The cardinality of the Supplementary Component.
 - ObjectClassTerm (mandatory): The Object Class of the Supplementary Component.
 - PropertyTerm (mandatory): The Property Term of the Supplementary Component.
 - PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the Supplementary Component.
 - PrimitiveType (mandatory): The primitive data type of the Supplementary Component.
 - UsageRule (optional, repetitive): A constraint that describes specific conditions that are 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.

7.5.2 Schema Construct

The unqualified data types 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.

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

```
<?xml version="1.0" encoding="utf-8"?>
<!-- ====== Unqualified Data Type Schema Module ===== -->
<!-- ===== Schema agency: UN/CEFACT ===== -->
<!-- ===== Schema version: 2.0 ===== -->
<!-- ===== Schema date: [SCHEMADATE] ===== -->
<!-- ... see intellectual property disclaimer ... -->
<xsd:schema targetNamespace=
    ... see namespace ...
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
<!-- ===== Imports ===== -->
<!-- ===== Type Definitions ===== -->
<!-- ===== Amount. Type ===== -->
<xsd:complexType name="AmountType">
    ... see type definition ...
</xsd:complexType>
...
</xsd:schema>
```

7.5.3 Namespace Scheme

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

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

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

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

```
<xsd:schema targetNamespace=
    "urn:un:unece:uncefact:data:draft:UnqualifiedDataType:1"
    xmlns:udt=
    "urn:un:unece:uncefact:data:draft:UnqualifiedDataType:1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
```

7.5.4 Imports and Includes

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

[R132] The **udt:UnqualifiedDataType** schema MUST only import the following schema modules:

- **ids:IdentifierList** schema modules
- **clm:CodeList** schema modules

2046 **Example 7-27: Imports**

```
2047     <!-- ===== Imports                               ==-->
2048     <!-- =====-->
2049     <!-- ===== Imports of Code Lists               ==-->
2050     <!-- =====-->
2051     <!-- =====-->
2052     <xsd:import namespace=
2053         "urn:un:unece:uncefact:codelist:standard:6:3055:D12A"
2054         schemaLocation="www.unece.org/uncefact/codelist/standard/UNECE_AgencyIdentificationCode_
D12A.xsd"/>
```

2055 **7.5.5 Type Definitions**

2056 Each unqualified data type is represented in the unqualified data type schema module as either a
2057 **xsd:complexType** or a **xsd:simpleType**. Unqualified data types are defined based on the core
2058 component types as specified in the CCTS.

-
- 2059 [R133] An unqualified data type MUST be defined for each approved primary and secondary
2060 representation terms identified in the CCTS Permissible Representation Terms table.
 - 2061 [R134] The name of each unqualified data type MUST be the dictionary entry name of the primary or
2062 secondary representation term, with the word 'Type' appended, the separators and spaces
2063 removed and approved abbreviations applied.
-

2064 In accordance with rules and principles in this document, the unqualified data type will be based on XSD built-
2065 in data types whenever the XSD built-in data type meets the functionality of all the supplementary
2066 components for that data type.

-
- 2067 [R135] For every unqualified data type whose supplementary components map directly to the properties
2068 of a XSD built-in data type, the unqualified data type MUST be defined as a named
2069 **xsd:simpleType** in the **udt:UnqualifiedDataType** schema module.
 - 2070 [R136] Every unqualified data type **xsd:simpleType** MUST contain one **xsd:restriction** element.
2071 This **xsd:restriction** element MUST include an **xsd:base** attribute that defines the specific
2072 XSD built-in data type required for the content component.
-

2073 When the unqualified data type does not directly map to an **xsd:simpleType** due to the supplementary
2074 components needing to be expressed, the unqualified data type will be defined as an **xsd:complexType**. If,
2075 however, some implementers want to use the simple type but others want to use the complex type, the
2076 unqualified data type should allow a choice between the two, using **xsd:choice**.

-
- 2077 [R137] For every unqualified data type whose supplementary components are not equivalent to the
2078 properties of a XSD built-in data type, the unqualified data type MUST be defined as an
2079 **xsd:complexType** in the **udt:UnqualifiedDataType** schema module.
 - 2080 [R138] Every unqualified data type **xsd:complexType** definition MUST contain one
2081 **xsd:simpleContent** element.
 - 2082 [R139] Every unqualified data type **xsd:complexType xsd:simpleContent** element MUST contain
2083 one **xsd:extension** element. This **xsd:extension** element must include an **xsd:base**
2084 attribute that defines the specific XSD built-in data type required for the content component.
 - 2085 [R204] When a combination of the complex and simple types are necessary to support business
2086 requirements, the element MUST be declared as an **xsd:complexType** with an **xsd:choice**
2087 between elements declared as the two different alternatives.
-

2088 **7.5.6 Attribute Declarations**

2089 Each core component supplementary component is declared as an attribute of the complex type. In certain
2090 circumstances, continually providing the attributes necessary to convey code and identifier list metadata for
2091 multiple repetitions of the same element may prove burdensome. The namespace scheme for code lists and
2092 identification scheme lists has been specifically designed to include some of the supplementary components
2093 for the CCTs **Code**, **Type** and **Identifier**, **Type**. If an implementation desires this metadata to be

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

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

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

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

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

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

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

[R143] 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.

[R144] 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.

Example 7-28: Type definitions of unqualified data types

```
<!-- ===== Type Definitions ===== -->
<!-- ===== Amount. Type ===== -->
<!-- ===== Binary Object. Type ===== -->

<xsd:complexType name="AmountType">
    <xsd:annotation>
        ... see annotation ...
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:decimal">
            <xsd:attribute name="currencyID"
                type="clm5ISO42173A:ISO3AlphaCurrencyCodeContentType" use="optional">
                <xsd:annotation>
                    ... see annotation ...
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="currencyCodeListVersionID" type="xsd:token" use="optional">
                <xsd:annotation>
                    ... see annotation ...
                </xsd:annotation>
            </xsd:attribute>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
<!-- ===== Binary Object. Type ===== -->
<!-- ===== Binary Object. Type ===== -->

<xsd:complexType name="BinaryObjectType">
    <xsd:annotation>
        ... see annotation ...
    </xsd:annotation>
    <xsd:simpleContent>
```

```

2153 <xsd:extension base="xsd:base64Binary">
2154   <xsd:attribute name="mimeCode" type="clmIANAMIMEMediaType: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.

-
- 2187 [R145] If the representation term of the supplementary component is 'Code' and validation is required,
2188 then the attribute representing this supplementary component MUST be based on the defined
2189 **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>
```

-
- 2202 [R146] If the representation term of the supplementary component is 'Identifier' and validation is
2203 required, then the attribute representing this supplementary component MUST be based on the
2204 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 MUST be present in the following pattern:
2240

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

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

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

2267 [R149] For every supplementary component `xsd:attribute` declaration a structured set of
2268 annotations MUST be present in the following pattern:
2269

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

- o Cardinality (mandatory): The cardinality of the Supplementary Component.
 - o ObjectClassTerm (mandatory): The Object Class of the Supplementary Component.
 - o PropertyTerm (mandatory): The Property Term of the Supplementary Component.
 - o PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the Supplementary Component.
 - o PrimitiveType (mandatory): The primitive data type of the Unqualified Data Type.
 - o UsageRule (optional, repetitive): A constraint that describes specific conditions that are 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
2331   ... 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>
```

7.6.3 Namespace Scheme

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

Example 7-35: Namespace name

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

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"
```

7.6.4 Imports and Includes

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

[R151] The **qdt:QualifiedDataType** schema module MUST import the **udt:UnqualifiedDataType** schema module.

[R205] The **qdt:QualifiedDataType** schema module MUST import all code list and identifier scheme schemas used in the module.

[Note]

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

7.6.5 Type Definitions

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

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

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

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

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

2378

Example 7-37: Type Definitions

```

2379      <!-- ===== Type Definitions ===== -->
2380      <!-- ===== Qualified Data Type based on DateTime Type ===== -->
2381      <!-- ===== Formatted_DateTime. Type ===== -->
2382      <!-- =====-->
2383      <!-- =====-->
2384      <!-- =====-->
2385      <!-- =====-->
2386      <!-- =====-->
2387      <!-- =====-->
2388      <!-- =====-->
2389      <!-- =====-->
2390      <!-- =====-->
2391      <!-- =====-->
2392      <!-- =====-->
2393      <!-- =====-->
2394      <!-- =====-->
2395      <!-- =====-->
2396      <!-- =====-->
2397      <!-- =====-->
2398      <!-- =====-->
2399      <!-- =====-->
2400      <!-- =====-->
2401      <!-- =====-->
2402      <!-- =====-->
2403      <!-- =====-->
2404      <!-- ===== Qualified Data Type based on Identifier. Type ===== -->
2405      <!-- =====-->
2406      <!-- ===== Country_ Identifier. Type ===== -->
2407      <!-- =====-->
2408      <!-- =====-->
2409      <!-- =====-->
2410      <!-- =====-->
2411      <!-- =====-->
2412      <!-- =====-->
2413      <!-- =====-->
2414      <!-- =====-->
2415      <!-- =====-->
2416      <!-- =====-->
2417      <!-- =====-->
2418      <!-- =====-->
2419      <!-- =====-->
2420      <!-- =====-->
2421      <!-- =====-->
2422      <!-- =====-->
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**.

2501 [R160] The qualified data type **xsd:complexType** definition **xsd:simpleContent** element MUST
2502 only restrict attributes declared in its base type, or MUST only restrict facets equivalent to
2503 inherited supplementary components.

2504

2505 **Example 7-41: Qualified Data Type Restricting an Identification Scheme**

```
2506 <xsd:complexType name="PartyIDType">  
2507   <xsd:annotation>  
2508     ... see annotation ...  
2509   </xsd:annotation>  
2510   <xsd:simpleContent>  
2511     <xsd:restriction base="udt:IDType">  
2512       <xsd:attribute name="schemeName" use="prohibited"/>  
2513       <xsd:attribute name="schemeAgencyName" use="prohibited"/>  
2514       <xsd:attribute name="schemeVersionID" use="prohibited"/>  
2515       <xsd:attribute name="schemeDataURI" use="prohibited"/>  
2516     </xsd:restriction>  
2517   </xsd:simpleContent>  
2518 </xsd:complexType>
```

2519 **7.6.7 Annotation**

2520 [R161] Every qualified data type definition MUST contain a structured set of annotations in the
2521 following sequence and pattern:
2522

- UniqueID (mandatory): The identifier that references a Qualified Data Type instance in a
2523 unique and unambiguous way.
- Acronym (mandatory): The abbreviation of the type of component. In this case the value will
2524 always be QDT.
- DictionaryEntryName (mandatory): The official name of the Qualified Data Type.
- Version (mandatory): An indication of the evolution over time of the Qualified Data Type
2525 instance.
- Definition (mandatory): The semantic meaning of the Qualified Data Type.
- PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the Qualified
2526 Data Type.
- DataTypeQualifierTerm (mandatory): A term that qualifies the Representation Term in order
2527 to differentiate it from its underlying Unqualified Data Type and other Qualified Data Type.
- PrimitiveType (mandatory): The primitive data type of the Qualified Data Type.
- BusinessProcessContextValue (optional, repetitive): The business process context for this
2528 Qualified Data Type is associated.
- GeopoliticalRegionContextValue (optional, repetitive): The geopolitical/region contexts for
2529 this Qualified Data Type.
- OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
2530 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
2531 Data Type.
- SupportingRoleContextValue (optional, repetitive): The supporting role context for this
2532 Qualified Data Type.
- SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for
2533 this Qualified Data Type.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are
2534 applicable to the Qualified Data Type.
- Example (optional, repetitive): Example of a possible value of a Qualified Data Type.

2535 **Example 7-42: Annotation of qualified data types**

```
2536   ... see type definition ...  
2537   <xsd:annotation>  
2538     <xsd:documentation xml:lang="en">  
2539       <ccts:UniqueID>UN02000133</ccts:UniqueID>  
2540       <ccts:Acronym>QDT</ccts:Acronym>  
2541       <ccts:DictionaryEntryName>Subject_ Code. Type</ccts:DictionaryEntryName>  
2542       <ccts:Version>1.0</ccts:Version>
```

```
2560      <ccts:Definition>A character string used to represent a subject code.  
2561      </ccts:Definition>  
2562      <ccts:PrimaryRepresentationTerm>Code</ccts:PrimaryRepresentationTerm>  
2563      <ccts:PrimitiveType>string</ccts:PrimitiveType>  
2564      <ccts:DataTypeQualifierTerm>Subject</ccts:DataTypeQualifierTerm>  
2565      </xsd:documentation>  
2566      </xsd:annotation>  
2567      ... see type definition ...
```

- 2568 [R162] For every supplementary component **xsd:attribute** declaration a structured set of
2569 annotations MUST be present in the following pattern:
2570 ○ UniqueID (optional): The identifier that references a Supplementary Component of a Core
2571 Component Type instance in a unique and unambiguous way.
2572 ○ Acronym (mandatory): The abbreviation of the type of component. In this case the value will
2573 always be SC.
2574 ○ DictionaryEntryName (mandatory): The official name of a Supplementary Component.
2575 ○ Definition (mandatory): The semantic meaning of a Supplementary Component.
2576 ○ Cardinality (mandatory): Indication whether the Supplementary Component Property
2577 represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Core
2578 Component Type.
2579 ○ ObjectClassTerm (mandatory): The Object Class Term of the associated Supplementary
2580 Component.
2581 ○ PropertyTerm (mandatory): The Property Term of the associated Supplementary Component.
2582 ○ PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the
2583 associated Supplementary Component.
2584 ○ PrimitiveType (mandatory): The Primitive Type of the associated Supplementary Component.
2585 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
2586 applicable to the Supplementary Component.

2587 7.7 Code Lists

2588 Codes are an integral component of any business to business information flow. Codes have been developed
2589 over time to facilitate the flow of compressed, standardized values that can be easily validated for correctness
2590 to ensure consistent data. In order for the XML instance documents to be fully validated by the parsers, any
2591 codes used within the XML document need to be available as part of the schema validation process. Many
2592 international, national and sectoral agencies create and maintain code lists relevant to their area. If required to
2593 be used within an information flow, these code lists will be stored in their own schema, and are referred to as
2594 external code lists. For example, many of the existing code lists that exist in the United Nations Code List
2595 (UNCL) will be stored as external code list schema for use within other UN/CEFACT XSD Schema.

2596 It should be noted that the use of enumerated code lists in messages is entirely optional. Great care should
2597 be taken when using them due to the issues in maintenance that can be created by their use.

- 2598 [R163] Each UN/CEFACT maintained code list MUST be defined in its own schema module.

2599 External code lists must be used when they exist in schema module form and when they can be directly
2600 imported into a schema module.

2601 UN/CEFACT may design and use an internal code list schema where an existing external code list schema
2602 needs to be extended, or where no suitable external code list schema exists. If a code list schema is created,
2603 it should be globally scoped and designed for reuse and sharing.

- 2604 [R164] Internal code list schema MUST NOT duplicate existing external code list schema when the
2605 existing ones are available to be imported.

2606 7.7.1 Schema Construct

2607 The code list schema module will follow the general pattern for all UN/CEFACT XSD schema modules.
2608 Following the generic module information, the body of the schema will consist of code list definitions of the
2609 following general form:
2610

2611 **Example 7-43: Structure of code lists**

```
2612 <?xml version="1.0" encoding="UTF-8"?>
2613 <!-- ===== 6Recommendation20 - Code List Schema Module ===== -->
2614 <!-- ===== -->
2615 <!-- ===== -->
2616 <!--
2617   Schema agency:      UN/CEFACT
2618   Schema version:    2.0
2619   Schema date:       [SCHEMADATE]
```

```
2620   Code list name:     Measurement Unit Common Code
2621   Code list agency:   UNECE
2622   Code list version: 3
```

```
2623   ... see intellectual property disclaimer ...
2624   -->
2625   <xsd:schema targetNamespace=" ... see namespace ...
2626     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2627     elementFormDefault="qualified" attributeFormDefault="unqualified">
2628   <!-- ===== -->
2629   <!-- ===== Root Element ===== -->
2630   <!-- ===== -->
2631   ... see root element declaration ...
2632   <!-- ===== -->
2633   <!-- ===== Type Definitions ===== -->
2634   <!-- ===== -->
2635   <!-- ===== Type Definition: Measurement Unit Common Code Content Type == -->
2636   <!-- ===== -->
2637   ... see type definition ...
2638 </xsd:schema>
```

2639 **7.7.2 Namespace Name for Code Lists**

2640 The namespace name for code list is unique in order to convey some of the supplementary component
2641 information rather than including them as attributes. Specifically, the UN/CEFACT namespace structure for a
2642 namespace name of a code list should be:

```
2643 urn:un:unece:uncefact:codelist:<status>:<Code List Agency Identifier|Code List
2644 Agency Name Text>:<Code List Identification Identifier|Code List Name
2645 Text>:<Code List Version Identifier>
```

2646 Where:

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

2654 [R165] The namespace names for code list schemas MUST have the following structure:

```
2655   urn:un:unece:uncefact:codelist:<status>:<Code List Agency
2656   Identifier|Code List Agency Name Text>:<Code List Identification.
2657   Identifier|Code List Name Text>:<Code List Version. Identifier>
```

2658 Where:

2659 codelist = this token identifying the schema as a code list
2660 status = a token identifying the standards status of this code list: draft | standard
2661 Code List Agency Identifier = identifies the agency that manages a code list. The default
2662 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.
2663 Code List Agency Name Text = the name of the agency that maintains the code list.
2664 Code List Identification Identifier = identifies a list of the respective corresponding codes. listID
2665 is only unique within the agency that manages this code list. Code List Name Text = the
2666 name of a list of codes.

2667 Code List Version Identifier = identifies the version of a code list.

2668 [R166] This rule was combined with [R165].

2669

2670 **Example 7-44: Namespace name of a code list with an agency and a code list identifier at draft status**

```
2672 "urn:un:unece:uncefact:codelist:draft:6:3403:D.04A"  
2673       where  
2674       6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2675                   the Code List. Agency. Identifier  
2676       3403 = UN/CEFACT data element tag for Name type code representing the  
2677                   Code List. Identification. Identifier  
2678       D.04A = the version of the UN/CEFACT directory
```

2679 **Example 7-45: Namespace name of proprietary code list at draft status**

```
2680 "urn:un:unece:uncefact:codelist:draft:Security_Initiative:Document_Security:1.2"  
2681       where  
2682       SecurityInitiative = the code list agency name of a responsible agency, which  
2683                           is not defined in UN/CEFACT data element 3055  
2684                           representing the Code List. Agency. Identifier  
2685       DocumentSecurity = the value for Code List. Name. Text  
2686       1.2 = the value for Code List. Version. Identifier
```

2687 **Example 7-46: Namespace name of a code list with an agency and a code list identifier at standard status**

```
2689 "urn:un:unece:uncefact:codelist:standard:6:3403:D.04A"  
2690       where  
2691       6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2692                   the Code List. Agency. Identifier  
2693       3403 = UN/CEFACT data element tag for Name status code representing the  
2694                   Code List. Identification. Identifier  
2695       D.04A = the version of the UN/CEFACT directory
```

2696 **Example 7-47: Namespace name of proprietary code list at standard status**

```
2697 "urn:un:unece:uncefact:codelist:standard:Security_Initiative:Document_Security:1.2"  
2698       where  
2699       SecurityInitiative = the code list agency name of a responsible agency, which  
2700                           is not defined in UN/CEFACT data element 3055  
2701                           representing the Code List. Agency. Identifier  
2702       DocumentSecurity = the value for Code List. Name. Text  
2703       1.2 = the value for Code List. Version. Identifier
```

2704 Versioning for code lists published by external organisations is outside of the control of UN/CEFACT. As with
2705 UN/CEFACT published code lists and identifier list schema the value of the Code List Version Identifier will
2706 follow the same rules as for versioning of other schema modules.

2707 **7.7.3 UN/CEFACT XSD Schema Namespace Token for Code Lists**

2708 A unique token will be defined for each namespace of code lists. The token representing the namespace for
2709 code lists should be constructed based on the identifier of the agency maintaining the code list and the
2710 identifier of the specific code list as issued by the maintenance agency except where there is no identifier.
2711 When there is no identifier, the name for the agency and/or code list should be used instead. This will typically
2712 be true when proprietary code lists are used. This method of token construction will provide uniqueness with
2713 a reasonably short token. When the code list is used for a qualified data type with a restricted set of valid
2714 code values, the qualified data type name is required to be used to distinguish one set of restricted values
2715 from another.

2716 The agency maintaining the code list will generally be either identified by the agency code as specified in data
2717 element 3055 in the UN/CEFACT Code List directory or the agency name if the agency does not have a code
2718 value in 3055. The identifier of the specific code list will generally be the data element tag of the
2719 corresponding list in the UN/CEFACT directory. If there is no corresponding data element, then the name of
2720 the code list will be used.

2721 In cases where the code list schema is a restricted set of values of a published code list schema, the code list
2722 schema will be associated with a qualified data type, and the name of the qualified data type will be included
2723 as part of the namespace token to ensure uniqueness from the unrestricted code list schema.

2724 [R167] Each UN/CEFACT maintained code list schema module MUST be represented by a unique token
2725 constructed as follows:

```
2726     clm[Qualified data type name]<Code List Agency Identifier|Code List  
2727     Agency Name Text><Code List Identification Identifier|Code List Name  
2728     Text>
```

2729 with any repeated words eliminated.

2730 **Example 7-48: Code list token with an agency and a code list identifier**

```
2731     The code list token for Name Type. Code is clm63403  
2732     where  
2733         6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2734             the Code List. Agency. Identifier  
2735         3403 = UN/CEFACT data element tag for Name status code representing  
2736             the Code List. Identification. Identifier
```

2738 **Example 7-49: Code list token for a qualified data type with an agency and code list identifiers**

```
2739     Code list token for Person_Name Type. Code is clmPersonNameType63403  
2740     where  
2741         PersonNameType = name of the qualified data type  
2742         6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2743             the Code List. Agency. Identifier  
2744         3403 = UN/CEFACT data element tag for Name status code representing  
2745             the Code List. Identification. Identifier
```

2746 **Example 7-50: Code list token for a proprietary code list**

```
2747     Code list token for a proprietary code list for Document Security is  
2748     clmSecurityInitiativeDocumentSecurity  
2749     where  
2750         SecurityInitiative = the code list agency name of a responsible agency, which  
2751             is not defined in UN/CEFACT data element 3055  
2752                 representing the Code List. Agency. Identifier  
2753         DocumentSecurity = the value for Code List. Name. Text
```

2754 Based on the constructs identified in the above examples, a namespace declaration for a code list would
2755 appear as shown in Example 7-51.

2756 **Example 7-51: Target namespace declaration for a code list**

```
2757     <xsd:schema  
2758         targetNamespace="urn:un:unece:uncefact:codelist:draft:6:4437:D.04A"  
2759         xmlns:clm64437="urn:un:unece:uncefact:codelist:draft:6:4437:D.04A"  
2760         xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2761         elementFormDefault="qualified" attributeFormDefault="unqualified">
```

2762 [Note]

2763 External developers are encouraged to follow the above construct rule when customizing
2764 schema for code lists to ensure that there is no namespace conflict.

2765 **7.7.4 Schema Location**

2766 Schema locations of code lists are typically defined as URL based URI schemes because of resolvability
2767 limitations of URN based URI schemes. However, UN/CEFACT XSD Schema of code lists use a URN
2768 based URI scheme for namespace declarations because persistence is considered more important than
2769 resolvability. In recognition of the need for resolvability of schema location, until such time as URNs become
2770 fully resolvable, UN/CEFACT will store schema of code lists in locations identified using a URL based URI
2771 scheme aligned with the URN based URI scheme used for the namespace declaration as follows:

```
2772     urn:un:unece:uncefact:codelist:<status>:<Code List. Agency Identifier|Code  
2773     List. Agency Name. Text>:<Code List. Identification. Identifier|Code List.  
2774     Name. Text>:<Code List. Version. Identifier>
```

2775 [R168] The structure for schema location of code lists MUST be:

2776 `.../codelist/<status>/<Code List Agency Identifier|Code List Agency`
2777 `Name Text>/<Code List Identification Identifier|Code List Name`
2778 `Text>_<Code List Version Identifier>.xsd`

2779 Where:

2780 schematype = a token identifying the type of schema module: codelist

2781 status = the status of the schema as: draft | standard

2782 Code List Agency Identifier = identifies the agency that manages a code list. The default
2783 agencies used are those from DE 3055. Code List Agency Name Text = the name of the
2784 agency that maintains the code list.

2785 Code List Identification Identifier = identifies a list of the respective corresponding codes. listID is
2786 only unique within the agency that manages this code list.

2787 Code List Name Text = the name of a list of codes.

2788 Code List Version Identifier = identifies the version of a code list.

2789 [R169] Each `xsd:schemaLocation` attribute declaration of a code list MUST contain a resolvable URL,
2790 and if an absolute path is used, it MUST also be persistent.

2791 [R170] This rule has been removed.

7.7.5 Imports and Includes

2793 UN/CEFACT Code List Schema Modules are standalone schema modules and will not import or include any
2794 other schema modules.

2795 [R171] Code List schema modules MUST not import or include any other schema modules.

7.7.6 Type Definitions

2797 [R172] Within each code list module one, and only one, named `xsd:simpleType` MUST be defined for
2798 the content component.

2799 [R173] The name of the `xsd:simpleType` MUST be the name of code list root element with the word
2800 'ContentType' appended.

Example 7-52: Simple type definition of code lists

```
2802 <!-- =====-->  
2803 <!-- ===== Type Definitions ===== -->  
2804 <!-- =====-->  
2805 <!-- ===== Type Definition: Party Role Type Code ===== -->  
2806 <!-- =====-->  
2807 <xsd:simpleType name="PartyRoleCodeContentType">  
2808     <xsd:restriction base="xsd:token">  
2809         <xsd:enumeration value="AA">  
2810             ... see enumeration ...  
2811         </xsd:enumeration>  
2812         ...  
2813         </xsd:restriction>  
2814 </xsd:simpleType>
```

2815 [R174] The `xsd:restriction` element base attribute value MUST be set to `xsd:token`.

2816 [R175] Each code in the code list MUST be expressed as an `xsd:enumeration`, where the
2817 `xsd:value` for the enumeration is the actual code value.

Example 7-53: Enumeration facet of code lists

```
2819 ... see type definition ...  
2820 <xsd:enumeration value="AA">  
2821     <xsd:annotation>  
2822         ... see annotation  
2823         </xsd:annotation>  
2824     </xsd:enumeration>  
2825     <xsd:enumeration value="AB">  
2826         <xsd:annotation>
```

```
2827           ... see annotation  
2828       </xsd:annotation>  
2829   </xsd:enumeration>  
2830 ...
```

2831 The purpose of the code list schema module is to define the list of allowable values (enumerations) that can
2832 appear within a particular element. Facet restrictions may be included in code lists if desired for additional
2833 edit check validation.

2834 **7.7.7 Element and Attribute Declarations**

2835 Each code list schema module will have a single **xsd:simpleType** defined. This single **xsd:simpleType**
2836 definition will have a **xsd:restriction** expression whose base is a XSD built-in data type. The
2837 **xsd:restriction** will be used to convey the content component enumeration value(s).

-
- 2838 [R176] For each code list a single root element MUST be globally declared.
 - 2839 [R177] The name of the code list root element MUST be the name of the code list following the
2840 naming rules as defined in section 5.3.
 - 2841 [R178] The code list root element MUST be of a type representing the actual list of code values.

2842 **Example 7-54: Root element declaration of code lists**

```
2843 <!-- =====-->  
2844 <!-- ===== Root Element ===== -->  
2845 <!-- =====-->  
2846 <xsd:element name="PartyRoleCode" type="clm63035:PartyRoleCodeContentType"/>
```

2847 The global declaration of a root element for each code list allows the use of code lists from different
2848 namespaces in a schema module when using **xsd:choice**.

2849 **Example 7-55: Usage of a choice of code lists**

```
2850 <xsd:complexType name="CalculationCurrencyCode">  
2851   <xsd:annotation>  
2852     ... see annotation ...  
2853   </xsd:annotation>  
2854   <xsd:choice>  
2855     <xsd:element ref="clm54217-N:CurrencyCode"/>  
2856     <xsd:element ref="clm54217-A:CurrencyCode"/>  
2857   </xsd:choice>  
2858 </xsd:complexType>
```

2859 **7.7.8 Extension and Restriction**

2860 Users of the UN/CEFACT library may identify any subset or superset they wish from a specific code list for
2861 their own trading community requirements by defining a qualified data type.

2862 Representation of a qualified data type of code lists could be

- 2863 ○ a combination of several individual code lists using **xsd:union**
- 2864 ○ a choice between several code lists, using **xsd:choice**
- 2865 ○ subsetting an existing code list using **xsd:restriction**

2866 Each of these can easily be accommodated in this syntax solution as required by the user's business
2867 requirements. Appendix D provides detailed examples of the various code list options.

2868 **7.7.9 Annotation**

2869 In order to facilitate a clear and unambiguous understanding of the list of allowable codes within an
2870 element, annotations will be provided for each enumeration to provide the code name and description.

-
- 2871 [R179] Each code list **xsd:enumeration** MUST contain a structured set of annotations in the
2872 following sequence and pattern:
 - Name (mandatory): The name of the code.
 - Description (optional): Descriptive information concerning the code.

2876 **Example 7-56: Annotation of codes**

```
2877   <xsd:enumeration value="AI">
2878     <xsd:annotation>
2879       <xsd:documentation xml:lang="en">
2880         <ccts:Name>Successful job applicant</ccts:Name>
2881         <ccts:Description>Person who has been chosen for a job. </ccts:Description>
2882       </xsd:documentation>
2883     </xsd:annotation>
2884   </xsd:enumeration>...
```

2885 **7.8 Identifier List Schema**

2886 When required, separate schema modules will be defined for identification schemes that have a token, and
2887 optionally a description, and that have the same functionality as a code list. In this way, XML instance
2888 documents containing these identifiers can be fully validated by the parsers. Other identifier schemes should
2889 be defined as a qualified or unqualified data type as appropriate.

2890 External identifier lists must be used when they exist in schema module form and when they can be directly
2891 imported into a schema module.

2892 UN/CEFACT may design and use an internal identifier list where an existing external identifier list needs to be
2893 extended, or where no suitable external identifier list exists. If an identifier list is created, the lists should be
2894 globally scoped and designed for reuse and sharing.

2895 It should be noted that the use of enumerated identifier lists in messages is entirely optional. Great care
2896 should be taken when using them due to the issues in maintenance that can be created by their use.

2897 [R180] Internal identifier lists schema MUST NOT duplicate existing external identifier list schema when
2898 the existing ones are available to be imported.

2899 [R181] Each UN/CEFACT maintained identifier list MUST be defined in its own schema module.

2900 **7.8.1 Schema Construct**

2901 The identifier list schema module will follow the general pattern for all UN/CEFACT XSD schema modules.
2902 Following the generic module information, the body of the schema will consist of identifier list definitions of the
2903 following general form:

2904 **Example 7-57: Structure of identifier lists**

```
2905   <?xml version="1.0" encoding="UTF-8"?>
2906   <!-- ===== Agency Identifier - Identifier List Schema Module ===== -->
2907   <!-- ===== Schema agency:      UN/CEFACT
2908   <!-- ===== Schema version:    2.0
2909   <!-- ===== Schema date:       [SCHEMADATE]
2910
2911   Identifier list name:          Agency Identifier
2912   Identifier list agency:        UNECE
2913   Identifier list version:       3
2914
2915   ... see intellectual property disclaimer ...
2916
2917   -->
2918   <xsd:schema targetNamespace=" ... see namespace ...
2919     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2920     elementFormDefault="qualified" attributeFormDefault="unqualified">
2921     <!-- ===== Root Element ===== -->
2922     <!-- ===== ...
2923       ... see root element declaration ...
2924     <!-- ===== Type Definitions ===== -->
2925     <!-- ===== Type Definition: Agency Identifier ===== -->
2926     <!-- ===== ...
2927       ... see type definition ...
2928     </xsd:schema>
```

7.8.2 Namespace Name For Identifier List Schema

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

```
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>
```

Where:

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

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

```
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>
```

Where:

status = the token identifying the publication status of this identifier scheme schema = draft|standard

identifierlist = this token identifying the schema as an identifier scheme

Identifier Scheme Agency Identifier = the identification of the agency that maintains the identification scheme.

Identifier Scheme Agency Name. Text = the name of the agency that maintains the identification list.

Identifier Scheme Identifier = the identification of the identification scheme.

Identifier Scheme Name. Text = the name of the identification scheme.

Identifier Scheme Version. Identifier = the version of the identification scheme.

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

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

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

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

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

Versioning for identifier list schemas published by external organisations is outside of the control of UN/CEFACT. As with UN/CEFACT published identifier list schema the value of the Identifier Scheme Version 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
```

3034 Where:
3035 schematype = a token identifying the type of schema module: identifierlist
3036 status = the status of the schema as: **draft | standard**
3037 Identifier Scheme. Agency Identifier = the identification of the agency that maintains the
3038 identification scheme.
3039 Identifier Scheme. Agency Name. Text = the name of the agency that maintains the
3040 identification scheme.
3041 Identifier Scheme. Identifier = the identification of the identification scheme.
3042 Identifier Scheme. Name. Text = the name of the identification scheme.
3043 Identifier Scheme. Version. Identifier = the version of the identification scheme.

3044 [R186] Each **xsd:schemaLocation** attribute declaration of an identifier list schema MUST contain a
3045 resolvable URL, and if an absolute path is used, it MUST also be persistent.

3046 [R187] This rule has been removed.

3047 **7.8.5 Imports and Includes**

3048 UN/CEFACT Identifier List Schema Modules are standalone schema modules and will not import or include
3049 any other schema modules.

3050 [R188] Identifier list schema modules MUST NOT import or include any other schema modules.

3051 **7.8.6 Type Definitions**

3052 A restriction has to be declared in order to define the content component (the simple type) as a restriction of
3053 the unqualified data type in order to comply with parser requirements. The restriction itself is the list of
3054 enumerations.

3055 [R189] Within each identifier list schema module one, and only one, named **xsd:simpleType** MUST
3056 be defined for the content component.

3057 [R190] The name of the **xsd:simpleType** MUST be the name of the identifier list root element with the
3058 word 'ContentType' appended.

3059 **Example 7-62: Simple type definition of an identifier list**

```
3060 <!-- ===== Type Definitions ===== -->
3061 <!-- =====-->
3062 <xsd:simpleType name="PaymentTermsDescriptionIdentifierContentType">
3063     <xsd:restriction base="xsd:token">
3064         <xsd:enumeration value="1">
3065             ... see enumeration ...
3066         </xsd:enumeration>
3067     </xsd:restriction>
3068 </xsd:simpleType>
```

3069 [R191] The **xsd:restriction** element base attribute value MUST be set to **xsd:token**.

3070 [R192] Each identifier in the identifier list MUST be expressed as an **xsd:enumeration**, where the
3071 **xsd:value** for the enumeration is the actual identifier value.

3072 **Example 7-63: Enumeration facet of an identifier list**

```
3073     ... see type defintion ...
3074     <xsd:enumeration value="1">
3075         <xsd:annotation>
3076             ... see annotation
3077         </xsd:annotation>
3078     </xsd:enumeration>
3079     <xsd:enumeration value="2">
3080         <xsd:annotation>
3081             ... see annotation
3082         </xsd:annotation>
3083     </xsd:enumeration>
3084     ...
```

3085 The purpose of the identifier list schema module is to define the list of allowable values (enumerations) that
3086 can appear within a particular element. Therefore, no other facet restrictions are allowed.

3087 [R193] Facets other than **xsd:enumeration** MUST NOT be used in the identifier list schema module.

3088 7.8.7 Attribute and Element Declarations

3089 Each identifier list schema module will have a single **xsd:simpleType** defined. This single
3090 **xsd:simpleType** definition will have a **xsd:restriction** expression whose base is a XSD built-in data
3091 type. The **xsd:restriction** will be used to convey the content component enumeration value(s).

3092 [R194] For each identifier list a single root element MUST be globally declared.

3093 [R195] The name of the identifier list root element MUST be the name of the identifier list following the
3094 naming rules as defined in section 5.3.

3095 [R196] The identifier list root element MUST be of a type representing the actual list of identifier values.

3096 Example 7-64: Root element declaration of identifier lists

```
3097 <!-- ===== -->  
3098 <!-- ===== Root Element ===== -->  
3099 <!-- ===== -->  
3100 <xsd:element name="PaymentTermsDescriptionIdentifier" type="ids64277:  
3101 PaymentTermsDescriptionIdentifierContentType"/>
```

3102 The global declaration of a root element for each identifier list allows the use of identifier lists from different
3103 namespaces in a schema module when using **xsd:choice**.

3104 Example 7-65: Usage of a choice of identifier lists

```
3105 <xsd:complexType name="CalculationCurrencyCode">  
3106     <xsd:annotation>  
3107         ... see annotation ...  
3108     </xsd:annotation>  
3109     <xsd:choice>  
3110         <xsd:element ref="clm54217-N:CurrencyCode"/>  
3111         <xsd:element ref="clm54217-A:CurrencyCode"/>  
3112     </xsd:choice>  
3113 </xsd:complexType>
```

3114 7.8.8 Extension and Restriction

3115 Users of the UN/CEFACT library may identify any subset or superset they wish from a specific identifier list for
3116 their own trading community requirements by defining a qualified data type.

3117 Representation of a qualified data type of identifier lists could be

- 3118 ○ a combination of several individual identifier lists using **xsd:union**
- 3119 ○ a choice between several identifier lists, using **xsd:choice**
- 3120 ○ subsetting an existing code list using **xsd:restriction**

3121 Each of these can easily be accommodated in this syntax solution as required by the user's business
3122 requirements. Appendix D provides detailed examples of the various identifier list options.

3123 XML declarations for using identifier lists in qualified data types are shown in the following examples.

3124 Example 7-66: Enumeration facet of identifier scheme

```
3125     ... see type definition ...  
3126     <xsd:enumeration value="AD">  
3127         <xsd:annotation>  
3128             ... see annotation ...  
3129         </xsd:annotation>  
3130     </xsd:enumeration>  
3131     <xsd:enumeration value="AE">  
3132         <xsd:annotation>  
3133             ... see annotation ...  
3134         </xsd:annotation>  
3135     </xsd:enumeration>  
3136     <xsd:enumeration value="AF">
```

```
3137      <xsd:annotation>
3138          ... see annotation ...
3139      </xsd:annotation>
3140  </xsd:enumeration>
```

3141 Example 7-67: Usage of only one identifier scheme

```
3142      <xsd:simpleType name="CountryIDType">
3143          <xsd:annotation>
3144              ... see annotation ...
3145          </xsd:annotation>
3146          <xsd:restriction base="ids53166:CountryIDContentType"/>
3147      </xsd:simpleType>
```

3148 Example 7-68: Usage of alternative identifier schemes

```
3149      <xsd:complexType name="GeopoliticalIDType">
3150          <xsd:annotation>
3151              ... see annotation ...
3152          </xsd:annotation>
3153          <xsd:choice>
3154              <xsd:element ref="ids53166:CountryCode"/>
3155              <xsd:element ref="ids53166-2:RegionCode"/>
3156          </xsd:choice>
3157      </xsd:complexType>
```

3158 7.8.9 Annotation

3159 In order to facilitate a clear and unambiguous understanding of the list of allowable identifiers within an
3160 element, annotations will be provided for each enumeration to provide the name, and optionally a description
3161 of, the identifier.

3162 [R197] Each **xsd:enumeration** MUST contain a structured set of annotations in the following
3163 sequence and pattern:

- 3164 o Name (mandatory): The name of the identifier.
- 3165 o Description (optional): Descriptive information concerning the identifier.

3166 Example 7-69: Annotation of Identifiers

```
3167      <xsd:enumeration value="1">
3168          <xsd:annotation>
3169              <xsd:documentation xml:lang="en">
3170                  <ccts:Name>Draft(s) drawn on issuing bank</ccts:Name>
3171                  <ccts:Description>Draft(s) must be drawn on the issuing
3172                      bank.</ccts:Description>
3173              </xsd:documentation>
3174          </xsd:annotation>
3175      </xsd:enumeration>
```

3176 8 XML Instance Documents

3177 In order to be UN/CEFACT conformant, an instance document must be valid against the relevant
3178 UN/CEFACT compliant XML schema. The XML instance documents should be readable and understandable
3179 by both humans and applications, and should enable reasonably intuitive interactions. It should represent all
3180 truncated tag names as described in section 7. A XPath navigation path should describe the complete
3181 semantic understanding by concatenating the nested elements. This navigation path should also reflect the
3182 meaning of each dictionary entry name of a BBIE or ASBIE.

3183 8.1 Character Encoding

3184 In conformance with ISO/IEC/ITU/UNECE Memorandum of Understanding Management Group (MOUMG)
3185 Resolution 01/08 (MOU/MG01n83) all UN/CEFACT XML will be instantiated using UTF. UTF-8 is the
3186 preferred encoding, but UTF-16 may be used where necessary to support other languages.

-
- 3187 [R198] All UN/CEFACT XML MUST be instantiated using UTF. UTF-8 should be used as the preferred
3188 encoding. If UTF-8 is not used, UTF-16 MUST be used.
-

3189 8.2 xsi:schemaLocation

3190 The **xsi:schemaLocation** and **xsi:noNamespaceLocation** attributes are part of the XML schema
3191 instance namespace (<http://www.w3.org/2001/XMLSchema-instance>). To ensure consistency, the token **xsi**
3192 will be used to represent the XML schema instance namespace.

-
- 3193 [R199] The **xsi** prefix MUST be used where appropriate for referencing **xsd:schemaLocation** and
3194 **xsd:noNamespaceLocation** attributes in instance documents.
-

3195 8.3 Empty Content

3196 Empty elements do not provide the level of assurance necessary for business information exchanges and as
3197 such, will not be used.

-
- 3198 [R200] UN/CEFACT conformant instance documents MUST NOT contain an element devoid of content.
3199 [R201] The **xsi:nil** attribute MUST NOT appear in any conforming instance.
-

3200 8.4 xsi:type

3201 The **xsi:type** attribute allows for substitution during an instantiation of a xml document. In the same way
3202 that substitution groups are not allowed, the **xsi:type** attribute is not allowed.

-
- 3203 [R202] The **xsi:type** attribute MUST NOT be used
-

3205 **Appendix A Related Documents**

3206 The following documents provided significant levels of influence in the development of this document:

- 3207 • UN/CEFACT Core Components Technical Specification, Part 8 of the ebXML Framework Version
3208 2.01 with corrigenda.
3209 • UN/CEFACT Core Components Business Document Assembly Technical Specification.

3210

Appendix B Overall Structure

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

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

B.1 XML Declaration

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

Example B-1: XML Declaration

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

B.2 Schema Module Identification and Intellectual Property Disclaimer

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

```
<!-- ===== -->
<!-- ===== Example - Schema Module Name ===== -->
<!-- ===== -->
<!--
      Schema agency:          UN/CEFACT
      Schema version:         2.0
      Schema date:            [SCHEMADATE]
-->
```

ECE draws attention to the possibility that the practice or implementation of its outputs (which include but are not limited to Recommendations, norms, standards, guidelines and technical specifications) may involve the use of a claimed intellectual property right.

Each output is based on the contributions of participants in the UN/CEFACT process, who have agreed to waive enforcement of their intellectual property rights pursuant to the UN/CEFACT IPR Policy (document ECE/TRADE/C/CEFACT/2010/20/Rev.2 available at http://www.unece.org/cefact/cf_docs.html or from the ECE secretariat). ECE takes no position concerning the evidence, validity or applicability of any claimed intellectual property right or any other right that might be claimed by any third parties related to the implementation of its outputs. ECE makes no representation that it has made any investigation or effort to evaluate any such rights.

Implementers of UN/CEFACT outputs are cautioned that any third-party intellectual property rights claims related to their use of a UN/CEFACT output will be their responsibility and are urged to ensure that their use of UN/CEFACT outputs does not infringe on an intellectual property right of a third party.

ECE does not accept any liability for any possible infringement of a claimed intellectual property right or any other right that might be claimed to relate to the implementation of any of its outputs.

-->

B.3 Schema Start Tag

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

- 3262 • Version
 3263 • Namespaces
 - targetNamespace attribute xmlns:xsd attribute
 - namespace declaration for current schema
 - namespace declaration for reusable ABIEs actually used in the schema
 - namespace declaration for unqualified data types actually used in the schema
 - namespace declaration for qualified data types actually used in the schema
 - namespace declaration for code lists actually used in the schema
 - namespace declaration for identifier schemes actually used in the schema
 - namespace declaration for CCTS
- 3272 • Form Defaults elementFormDefault attributeFormDefault
- 3273 • Others
 - other schema attributes with schema namespace
 - other schema attributes with non-schema namespace

Example B-3: XML Schema Start Tag

```

3277 <xsd:schema
3278   xmlns:ccts="urn:un:unece:uncefact:documentation:standard:CoreComponentsTechnicalSpecificat
3279   ion:2"
3280   xmlns:clm6Recommendation20="urn:un:unece:uncefact:codelist:standard:6:Recommendation20:8"
3281   xmlns:clm60133="urn:un:unece:uncefact:codelist:standard:6:0133:40106"
3282   xmlns:clm5ISO42173A="urn:un:unece:uncefact:codelist:standard:5:ISO42173A:2012-08-31"
3283   xmlns:ids5ISO316612A="urn:un:unece:uncefact:identifierlist:standard:5:ISO316612A:SecondEdi
3284   tion2006VI-13"
3285   xmlns:clmIANAMIMEMediaType="urn:un:unece:uncefact:codelist:standard:IANA:MIMEMediaType:201
3286   3-01-03"
3287   xmlns:clmIANACharacterSetCode="urn:un:unece:uncefact:codelist:standard:IANA:CharacterSetCo
3288   de:2013-01-08" xmlns:clm63055="urn:un:unece:uncefact:codelist:standard:6:3055:D12A"
3289   xmlns:udc="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:13"
3290   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3291   targetNamespace="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:13"
3292   elementFormDefault="qualified"
3293   attributeFormDefault="unqualified"
3294   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

```

3300 <!-- ===== -->
3301 <!-- ===== Include ===== -->
3302 <!-- ===== -->
3303 <!-- ===== Inclusion of internal ABIE ===== -->
3304 <!-- ===== -->
3305 <!-- ===== -->
3306   <xsd:include
3307     namespace="urn:un:unece:uncefact:data:standard:CIIAggregateBusinessInformationEntity
3308     :1"
3309     schemaLocation="http://www.unece.org/uncefact/data/standard/CIIAggregateBusinessInformationE
3310     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

3318

Example B-5: Imports

```

3319      <!-- =====>
3320      <!-- ===== Imports ===== -->
3321      <!-- =====>
3322      <!-- ===== Import of Unqualified Data Type ===== -->
3323      <!-- =====>
3324      <xsd:import namespace="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:14"
3325          schemaLocation="http://www.unece.org/uncefact/data/standard/UnqualifiedDataType_14.xsd" />
3326      <!-- =====>
3327      <!-- ===== Import of Qualified Data Type ===== -->
3328      <!-- =====>
3329      <xsd:import namespace="urn:un:unece:uncefact:data:standard:QualifiedDataType:14"
3330          schemaLocation="http://www.unece.org/uncefact/data/standard/QualifiedDataType_14.xsd"/>
3331      >
3332      <!-- =====>
3333      <!-- ===== Import of Reusable Aggregate Business Information Entity Schema Module -->
3334      <!-- =====>
3335      <xsd:import
3336          namespace="urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEn
3337          tity:13" schemaLocation="ReusableAggregateBusinessInformationEntity_13p0.xsd"/>
```

3338

B.6 Root Element

3339

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

3341

Example B-6:

```

3342      <!-- =====>
3343      <!-- ===== Element Declarations ===== -->
3344      <!-- =====>
3345      <!-- ===== Root element ===== -->
3346      <!-- =====>
3347      <xsd:element name="[ELEMENTNAME]" type="[TOKEN] : [TYPENAME]" />
3348      <!-- =====>
3349      <!-- ===== Global Element Declarations ===== -->
3350      <!-- =====>
3351      <xsd:element name="[ELEMENTNAME]" type="[TOKEN] : [TYPENAME]" />
3352      <!-- =====>
```

3353

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

3354

Example B-7:

```

3355      <!-- =====>
3356      <!-- ===== Root element ===== -->
3357      <!-- =====>
3358      <xsd:element name="CrossIndustryInvoice" type="rsm: CrossIndustryInvoiceType">
3359          <xsd:annotation>
3360              <xsd:documentation>
3361                  <ccts:UniqueID>CII</ccts:UniqueID>
3362                  <ccts:Acronym>RSM</ccts:Acronym>
3363                  <ccts:Name>CrossIndustryInvoice</ccts:Name>
3364                  <ccts:Version>1.0</ccts:Version>
3365                  <ccts:Definition>A message used as a request for payment, or modification of
3366                      a request for payment, for the supply of goods or services ordered,
3367                      delivered, received, consumed.</ccts:Definition>
3368                  <ccts:BusinessProcessContextValue>PurchaseOrder</ccts:BusinessProcessContextValue>
3369                  <ccts:GeopoliticalOrRegionContextValue>In All
3370                      Contexts</ccts:GeopoliticalOrRegionContextValue>
3371                  <ccts:OfficialConstraintContextValue>None</ccts:OfficialConstraintContextValue>
3372                  <ccts:ProductContextValue>In All Contexts</ccts:ProductContextValue>
3373                  <ccts:IndustryContextValue>In All Contexts</ccts:IndustryContextValue>
3374                  <ccts:BusinessProcessRoleContextValue>In All
3375                      Contexts</ccts:BusinessProcessRoleContextValue>
3376                  <ccts:SupportingRoleContextValue>In All
3377                      Contexts</ccts:SupportingRoleContextValue>
3378                  <ccts:SystemCapabilitiesContextValue>In All
3379                      Contexts</ccts:SystemCapabilitiesContextValue>
3380                  </ccts:Documentation>
3381          </xsd:annotation>
3382      </xsd:element>
```

3384

Example B-8: Global elements

```

3385      <!-- ===== -->
3386      <!-- ===== Global element ===== -->
3387      <!-- ===== -->
3388      <xsd:element name="AdministrativeCountrySubDivision"
3389      type="ram:AdministrativeCountrySubDivisionType"/>
3390          <xsd:annotation>
3391              <xsd:documentation>
3392                  <ccts:UniqueID>UN01009362</ccts:UniqueID>
3393                  <ccts:Acronym>ABIE</ccts:Acronym>
3394                  <ccts:DictionaryEntryName>Administrative_ Country Subdivision.
3395                  Details</ccts:DictionaryEntryName>
3396                  <ccts:Version>1.0</ccts:Version>
3397                  <ccts:Definition>An area which is an administrative sub-division within a
3398                  country, such as a state, a county, a canton or a province.</ccts:Definition>
3399                  <ccts:ObjectClassTerm>Country Subdivision</ccts:ObjectClassTerm>
3400                      <ccts:QualifierTerm>Administrative</ccts:QualifierTerm>
3401                  </xsd:documentation>
3402          </xsd:annotation>
3403      </xsd:element>

```

3404

B.7 Type Definitions

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

3407

Example B-9: Type Definitions

```

3408      <!-- ===== -->
3409      <!-- ===== Type Definitions ===== -->
3410      <!-- ===== -->
3411      <!-- ===== Type Definition: Administrative SubDivision type ===== -->
3412      <!-- ===== -->
3413      <xsd:complexType name="AdministrativeCountrySubDivisionType">
3414          <xsd:annotation>
3415              <xsd:documentation xml:lang="en">
3416                  <ccts:UniqueID>UN01009362</ccts:UniqueID>
3417                  <ccts:Acronym>ABIE</ccts:Acronym>
3418                  <ccts:DictionaryEntryName>Administrative_ Country Subdivision.
3419                  Details</ccts:DictionaryEntryName>
3420                  <ccts:Version>1.0</ccts:Version>
3421                  <ccts:Definition>An area which is an administrative subdivision within a
3422                  country, such as a state, a county, a canton or a
3423                  province.</ccts:Definition>
3424                  <ccts:ObjectClassTerm>Country Subdivision</ccts:ObjectClassTerm>
3425                  <ccts:ObjectClassTermQualifier>Administrative</ccts:ObjectClassTermQualifier>
3426          </xsd:documentation>
3427      </xsd:annotation>
3428      <xsd:sequence>
3429          <xsd:element name="ID" type="udt:IDType ">
3430              <xsd:annotation>
3431                  <xsd:documentation xml:lang="en">
3432                      <ccts:UniqueID>UN01009363</ccts:UniqueID>
3433                      <ccts:Acronym>BBIE</ccts:Acronym>
3434                      <ccts:DictionaryEntryName>Administrative_ Country Subdivision.
3435                      Identification. Identifier</ccts:DictionaryEntryName>
3436                      <ccts:Version>1.0</ccts:Version>
3437                      <ccts:Definition>The identifier for this administrative country
3438                      subdivision.</ccts:Definition>
3439                      <ccts:Cardinality>1</ccts:Cardinality>
3440                      <ccts:ObjectClassTerm>Country Subdivision</ccts:ObjectClassTerm>
3441                      <ccts:ObjectClassTermQualifier>Administrative</ccts:ObjectClassTermQualifier>
3442          <ccts:PropertyTerm>Identification</ccts:PropertyTerm>
3443          <ccts:PrimaryRepresentationTerm>Identifier</ccts:PrimaryRepresentationTerm>
3444          </xsd:documentation>
3445      </xsd:annotation>
3446      </xsd:element>
3447      <xsd:element name="Description" type="udt:TextType" minOccurs="0
3448      ">
3449          <xsd:annotation>

```

```

3452     <xsd:documentation xml:lang="en">
3453         <cccts:UniqueID>UN01009364</cccts:UniqueID>
3454         <cccts:Acronym>BBIE</cccts:Acronym>
3455         <cccts:DictionaryEntryName>Administrative_Country Subdivision. Description.
3456         Text</cccts:DictionaryEntryName>
3457         <cccts:Version>1.0</cccts:Version>
3458         <cccts:Definition>The textual description for this
3459             administrative country subdivision.</cccts:Definition>
3460             <cccts:Cardinality>0..1</cccts:Cardinality>
3461             <cccts:ObjectClassTerm>Country Subdivision</cccts:ObjectClassTerm>
3462             <cccts:ObjectClassTermQualifier>Administrative</cccts:ObjectClassTermQualifier>
3463             <cccts:PropertyTerm>Description</cccts:PropertyTerm>
3464             <cccts:PrimaryRepresentationTerm>Text</cccts:PrimaryRepresentationTerm>
3465         </xsd:documentation>
3466     </xsd:annotation>
3467 </xsd:element>

```

3468 Example B-10: Complete Structure

```

3469 <?xml version="1.0" encoding="UTF-8"?>
3470 <!-- ===== [SCHEMA MODULE TYPE] Schema Module ===== -->
3471 <!-- ===== [SCHEMA AGENCY NAME] Schema Version ===== -->
3472 <!-- ===== [DATE OF SCHEMA] -->
3473 <!--
3474     Schema agency:          [SCHEMA AGENCY NAME]
3475     Schema version:        [SCHEMA VERSION] Schema
3476     date:                  [DATE OF SCHEMA]
3477
3478     [Code list name:]      [NAME OF CODE LIST] [Code
3479     list agency:]         [CODE LIST AGENCY] [Code
3480     list version:]        [VERSION OF CODE LIST]
3481     [Identifier list name:] [NAME OF IDENTIFIER LIST]
3482     [Identifier list agency:] [IDENTIFIER LIST AGENCY]
3483     [Identifier list version:] [VERSION OF IDENTIFIER LIST]
3484
3485 ECE draws attention to the possibility that the practice or implementation of its
3486 outputs (which include but are not limited to Recommendations, norms, standards,
3487 guidelines and technical specifications) may involve the use of a claimed intellectual
3488 property right.
3489
3490 Each output is based on the contributions of participants in the UN/CEFACT process, who
3491 have agreed to waive enforcement of their intellectual property rights pursuant to the
3492 UN/CEFACT IPR Policy (document ECE/TRADE/C/CEFACT/2010/20/Rev.2 available at
3493 http://www.unece.org/cefact/cf\_docs.html or from the ECE secretariat). ECE takes no
3494 position concerning the evidence, validity or applicability of any claimed intellectual
3495 property right or any other right that might be claimed by any third parties related to
3496 the implementation of its outputs. ECE makes no representation that it has made any
3497 investigation or effort to evaluate any such rights.
3498
3499 Implementers of UN/CEFACT outputs are cautioned that any third-party intellectual
3500 property rights claims related to their use of a UN/CEFACT output will be their
3501 responsibility and are urged to ensure that their use of UN/CEFACT outputs does not
3502 infringe on an intellectual property right of a third party.
3503
3504 ECE does not accept any liability for any possible infringement of a claimed
3505 intellectual property right or any other right that might be claimed to relate to the
3506 implementation of any of its outputs.-->
3507
3508 <xsd:schema
3509 targetNamespace="urn:un:unece:uncefact:data:draft:[MODULENAME]:[VERSION]"
3510 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3511 ... FURTHER NAMESPACES ...
3512
3513     elementFormDefault="qualified" attributeFormDefault="unqualified">
3514 <!-- ===== -->
3515 <!-- ===== Include ===== -->
3516 <!-- ===== Inclusion of [TYPE OF MODULE] ===== -->
3517 <!-- ===== -->
3518     <xsd:include namespace="..." schemaLocation="..."/>
3519 <!-- ===== -->
3520 <!-- ===== Imports ===== -->
3521 <!-- ===== -->
3522 <!-- ===== Import of [TYPE OF MODULE] ===== -->
3523 <!-- ===== -->
3524     <xsd:import namespace="..." schemaLocation="..."/>
3525 <!-- ===== -->

```

```
3520      <!-- ===== Element Declarations          ===== -->
3521      <!-- =====-->
3522      <!-- ===== Root element           ===== -->
3523      <!-- =====-->
3524      <xsd:element name="[ELEMENTNAME]" type="[TOKEN] : [TYPENAME]">
3525      <!-- =====-->
3526      <!-- ===== Global Element Declarations   ===== -->
3527      <!-- =====-->
3528      <xsd:element name="[ELEMENTNAME]" type="[TOKEN] : [TYPENAME]">
3529      <!-- =====-->
3530      <!-- ===== Type Definitions        ===== -->
3531      <!-- =====-->
3532      <!-- ===== Type Definition: [TYPE]     ===== -->
3533      <!-- =====-->
3534
3535      <xsd:complexType name="[TYPENAME]">
3536          <xsd:restriction base="xsd:token">
3537              ... see type definition ....
3538          </xsd:restriction>
3539      </xsd:complexType>
```

3539

3540 **Appendix C BPS Approved Acronyms and Abbreviations**

3541 The following constitutes a list of BPS approved acronyms and abbreviations which must be used within tag
3542 names when these words are part of the dictionary entry name:

3543 **ID – Identifier**

3544 **URI – Uniform Resource Identifier**

3545

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"/>
  <!-- Message 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.

3603 Within the **xsd:complexType** of **ProductType**, five local elements are declared. Each of these elements
3604 represent one of the five different code list options.

3605 D.1.1 Referencing a Predefined Standard Code List in and Unqualified Data 3606 Type

3607 In the Code Use Example Schema, the element **TotalAmount** is declared as:

```
3608 <xsd:element name="TotalAmount" type="udt:AmountType"/>
```

3609 As shown in the element declaration, **TotalAmount** is of the CCTS unqualified data type **udt:AmountType**
3610 which has been defined in the UN/CEFACT unqualified data type schema module (See Section 7.6). The
3611 **udt:AmountType** declaration in the unqualified schema module is as follows:

```
3612 <xsd:schema  
3613   targetNamespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:1"  
3614   xmlns:clm54217="urn:un:unece:uncefact:codelist:draft:5:4217:2001" ...  
3615   elementFormDefault="qualified" attributeFormDefault="unqualified">  
3616     <!-- =====-->  
3617     <!-- ===== Imports ===== -->  
3618     <!-- =====-->  
3619     <!-- ===== Imports of Code Lists ===== -->  
3620     <!-- =====-->  
3621     <!-- ===== Imports of Code Lists ===== -->  
3622     <xsd:import namespace="urn:un:unece:uncefact:codelist:draft:5:4217:2001"  
3623       schemaLocation=" http://www.unece.org/uncefact/codelist/draft/5/4217_2001_.xsd "/>  
3624     <!-- =====-->  
3625     <!-- ===== Type Definitions ===== -->  
3626     <!-- =====-->  
3627     <!-- ===== Amount. Type ===== -->  
3628     <!-- =====-->  
3629     <xsd:complexType name="AmountType">  
3630       <xsd:simpleContent>  
3631         <xsd:extension base="xsd:decimal">  
3632           <xsd:attribute name="currencyID" type="clm54217:CurrencyCodeContentType"  
3633             use="required"/>  
3634           </xsd:extension>  
3635         </xsd:simpleContent>  
3636     </xsd:complexType>
```

3636 This **udt:AmountType** has attributes declared that represent the supplementary components defined in
3637 CCTS for this data type. These attributes include **currencyCode** for the supplementary component of
3638 **Amount. Currency. Code**. This **currencyCode** attribute is declared to be of the **xsd:simpleType**
3639 **clm54217:CurrencyCodeContentType**. The **clm54217:CurrencyCodeContentType** has been
3640 declared in the code list schema module for ISO Currency Codes, and the allowed code values for the
3641 **currencyCode** attribute have been defined as enumeration facets in the
3642 **clm54217:CurrencyCodeContentType** type definition.

3643 An extract of the code list schema module for ISO Currency Codes is as follows:

```
3644   <!-- =====-->  
3645   <!-- ===== Root Element Declarations ===== -->  
3646   <!-- =====-->  
3647   <xsd:element name="CurrencyCode" type="clm54217:CurrencyCodeContentType"/>  
3648   <!-- =====-->  
3649   <!-- ===== Type Definitions ===== -->  
3650   <!-- =====-->  
3651   <!-- ===== Code List Type Definition: Country Codes ===== -->  
3652   <!-- =====-->  
3653   <xsd:simpleType name="CurrencyCodeContentType">  
3654     <xsd:restriction base="xsd:token">  
3655       <xsd:enumeration value="AED">  
3656         <xsd:annotation>  
3657           <xsd:documentation>  
3658             <CodeName>Dirham</CodeName>  
3659           </xsd:documentation>  
3660         </xsd:annotation>  
3661       </xsd:enumeration>  
3662       <xsd:enumeration value="AFN">  
3663         <xsd:annotation>  
3664           <xsd:documentation>
```

```

3665             <CodeName>Afghani</CodeName>
3666         </xsd:documentation>
3667     </xsd:annotation>
3668     </xsd:enumeration>
3669   </xsd:restriction>
3670 </xsd:simpleType>
3671 </xsd:schema>

```

3672 The **currencyCode** attribute has a fixed value of ISO 4217 Currency Code as defined in CCTS. Thus, only
 3673 code values from this code list are allowed in a CEFACt conformant instance document. In such an instance
 3674 document, actual conveyance of a currency code value would be represented as:

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

3676 It should be noted that when using this option, no information about the code list being used is carried in the
 3677 instance document as this information is already defined in the underlying XML schema.

3678 D.1.2 Referencing Any Code List Using the Unqualified Data Type

3679 udt:CodeType

3680 The second element in this example message – **TaxCurrencyCode** – is of the unqualified data type
 3681 **udt:CodeType**.

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

3683 This **udt:CodeType** data type includes a number of supplementary components required in order to
 3684 uniquely identify the code list to be used for validation.

3685 The **udt:CodeType** is declared in the unqualified schema module as:

```

3686 <xsd:complexType name="CodeType">
3687   <xsd:simpleContent>
3688     <xsd:extension base="xsd:token">
3689       <xsd:attribute name="listID" type="xsd:token" use="optional"/>
3690       <xsd:attribute name="listName" type="xsd:string" use="optional"/>
3691       <xsd:attribute name="listAgencyID" type="xsd:token" use="optional"/>
3692       <xsd:attribute name="listAgencyName" type="xsd:string" use="optional"/>
3693       <xsd:attribute name="listVersionID" type="xsd:token" use="optional"/>
3694       <xsd:attribute name="listURI" type="xsd:anyURI" use="optional"/>
3695     </xsd:extension>
3696   </xsd:simpleContent>
3697 </xsd:complexType>

```

3698 When the **udt:CodeType** is used, either the **listURI** (which will point uniquely to the code list) should be
 3699 used, or a combination of the other attributes should be used. Thus, it is possible to refer to the code list
 3700 relevant attributes either by the specific attributes for the explicit display of supplementary components, or
 3701 by the list URI in which the value is based on the namespace name conventions, such as:
 3702 **urn:un:unece:uncefact:codelist:draft:5:4217:2001**.

3703 The association to the specific namespace must be defined during runtime. In an instance document this
 3704 element could be represented as:

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

3707 or

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

3710 It should be noted that when applying this option, validation of code values in the instance document will not
 3711 be done by the XML parser.

3712 D.1.3 Referencing a Predefined Code List by Declaring a Specific Qualified 3713 Data Type

3714 The third element in our example message **ChangeCurrencyCode** is based on the qualified data type
 3715 **qdt:CurrencyCodeType**.

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

3717 The **qdt:CurrencyCodeType** would be defined in the qualified data type schema module as:

```
3718 <xsd:simpleType name="CurrencyCodeType">
3719   <xsd:restriction base="clm54217-A:CurrencyCodeContentType"/>
3720 </xsd:simpleType>
```

3721 This means that the value of the **ChangeCurrencyCode** element can only have code values from the
3722 identified ISO 4217 code list. In an instance document this element would be represented as:

```
3723 <ChangeCurrencyCode>AED</ChangeCurrencyCode>
```

3724 It should be noted that when using this option no information about the code list to be used is carried in the
3725 instance document as this is already defined in the XML schema.

D.1.4 Choosing or Combining Values from Different Code Lists

3726 The fourth option is to chose or combine values from diverse code lists by using either the **xsd:choice** or
3727 **xsd:union** elements.

D.1.4.1 Choice

3730 In the Code Use Example Schema, the element **CalculationCurrencyCode** is declared as:

```
3731 <xsd:element name="CalculationCurrencyCode" type="qdt:CalculationCurrencyCodeType"/>
```

3732 The **CalculationCurrencyCode** element is of qualified data type
3733 **qdt:CalculationCurrencyCodeType**.

3734 The **qdt:CalculationCurrencyCodeType** is defined in the qualified data type module as:

```
3735 <xsd:complexType name="CalculationCurrencyCodeType">
3736   <xsd:choice>
3737     <xsd:element ref="clm54217-N:CurrencyCode"/>
3738     <xsd:element ref="clm54217-A:CurrencyCode"/>
3739   </xsd:choice>
3740 </xsd:complexType>
```

3741 The **xsd:choice** element provides a choice of values from either the **clm54217-N:CurrencyCode** or from
3742 **clm54217-A:CurrencyCode**. The schema module for **clm54217-A:CurrencyCode** is the same as the
3743 one used in section 9.1.1 above. The sample schema module for **clm54217-N:CurrencyCode** is as
3744 follows:

Example D-2: Sample clm54217-N:CurrencyCode Schema Module:

```
3745 <!-- ===== Root Element Declarations ===== -->
3746 <!-- ===== Type Definitions ===== -->
3747 <!-- ===== Code List Type Definition: 4217-N Currency Codes ===== -->
3748 <!-- =====-->
3749 <xsd:element name="CurrencyCode" type="clm54217-N:CurrencyCodeContentType"/>
3750 <!-- =====-->
3751 <!-- =====-->
3752 <!-- ===== Code List Type Definition: 4217-N Currency Codes ===== -->
3753 <!-- =====-->
3754 <xsd:simpleType name="CurrencyCodeContentType">
3755   <xsd:restriction base="xsd:token">
3756     <xsd:enumeration value="840">
3757       <xsd:annotation>
3758         <xsd:documentation>
3759           <CodeName>US Dollar</CodeName>
3760         </xsd:documentation>
3761       </xsd:annotation>
3762     </xsd:enumeration>
3763     <xsd:enumeration value="978">
3764       <xsd:annotation>
3765         <xsd:documentation>
3766           <CodeName>Euro</CodeName>
3767         </xsd:documentation>
3768       </xsd:annotation>
3769     </xsd:enumeration>
3770   </xsd:restriction>
3771 </xsd:simpleType>
3772 </xsd:schema>
```

3773 This **xsd:choice** option allows for the use of code values from different pre-defined code lists in the
3774 instance document. The specific code list being used in the instance document will be represented by the
3775 namespace prefix (**clm54217-A** or **clm54217-N**) being used for the namespace declaration of the imported
3776 code list and for the **CurrencyCode** element:

```
3777 <PurchaseOrder ... xmlns:clm54217-N="urn:un:unece:uncefact:codelist:draft:5:4217-N:2001"  
3778 ...>  
3779     <CalculationCurrencyCode>  
3780         <clm54217-N:CurrencyCode>840</clm54217-N:CurrencyCode>  
3781     </CalculationCurrencyCode>  
3782     ...  
3783 </PurchaseOrder>
```

3784 The namespace prefix unambiguously identifies to the recipient of the instance from which code list each
3785 code value is defined.

3786 **D.1.4.2 Union**

3787 The **xsd:union** code list approach is similar to that for the **xsd:choice** approach in that multiple code
3788 lists are being used. The element declaration in the schema would be identical to that for choice in that the
3789 element **CalculationCurrencyCode** is still based on the qualified data type
3790 **qdt:CalculationCurrencyCodeType**.

```
3791 <xsd:element name="CalculationCurrencyCode" type="qdt:CalculationCurrencyCodeType"/>
```

3792 The difference is that the **qdt:calculationCurrencyCodeType** would be defined in the qualified data
3793 type module using an **xsd:union** element rather than an **xsd:choice** element:

```
3794     <xsd:simpleType name="CalculationCurrencyCodeType">  
3795         <xsd:union memberTypes="clm54217-N:CurrencyCodeContentType clm54217-  
3796             A:CurrencyCodeContentType"/>  
3797     </xsd:simpleType>
```

3798 Here the declaration enables the instance to select a choice of values from either the **clm54217-**
3799 **N:CurrencyCodeContentType** or from the **clm54217-A:CurrencyCodeContentType**. The code list
3800 schema module for **clm54217-A:CurrencyCodeContentType** is the same as the one used in Section
3801 D.1.1 above. The code list schema module for **clm54217-N:CurrencyCodeContentType** is the same
3802 as the one used in Section D.1.4.1.

3803 This **xsd:union** option allows for the use of code values from different pre-defined code lists in the
3804 instance document. The code lists must be imported once in the XML schema module and must be shown
3805 once in the XML instance. The specific code list will be represented by the namespace prefix (**clm54217-A**
3806 or **clm54217-N**), but unlike the choice option, the element in the instance document will not have the
3807 specific code list token conveyed as the first part of the element name. The recipient of the instance does
3808 not know unambiguously in which code list each code value is defined. This is because a reference to the
3809 specific code lists comes from different code list schema modules, such as, **clm54217- N** and **clm54217-**
3810 **A**.

3811 In an instance document this element could be represented as:

```
3812 <PurchaseOrder >  
3813     ...  
3814     <CalculationCurrencyCode>840</CalculationCurrencyCode>  
3815     ...  
3816 </PurchaseOrder>
```

3817 The advantage of the **xsd:union** approach is that attributes can make use of these code lists. For example,
3818 it may make sense for an implementation to standardize across the board on two currency code lists and
3819 have those apply to all of the data types, like **udt:AmountType** and its **currencyID** attribute.

3820 **D.1.5 Restricting Allowed Code Values**

3821 This option is used when it is desired to reduce the number of allowed code values from an existing code list.
3822 For example, a trading partner community may only recognize certain code values from the ISO 4217
3823 Currency Code list. To accomplish this, three options exist:

- Use **xsd:substitutionGroup** to replace the simple type that conveys the enumerated list of
3825 codes

- 3826 • Use **xsd:redefine** to replace the simple type that conveys the enumerated list of codes
 3827 • Create a new **xsd:simpleType** with the restricted set of value declarations

3828 The **xsd:substitutionGroup** and **xsd:redefine** features are specifically prohibited in the UN/CEFACT
 3829 XML NDR due to issues associated with authentication, non-repudiation, ease of understanding, and tool
 3830 support. Accordingly, when a user community wishes to restrict the allowed code values expressed in an
 3831 existing schema, a new qualified datatype will be created in the QDT schema module, a new restricted
 3832 codelist schema module will be created, and a new **xsd:simpleType** will be defined. This new
 3833 **xsd:simpleType** will contain a complete list of allowed enumerations.

3834 In the example in section D.1.1, a **CurrencyID** element was declared and this element was of the
 3835 **xsd:simpleType qdt:CurrencyCodeContentType** defined for currency code:

3836 If we wished to restrict the allowed values of **qdt:CurrencyCodeType**, we will have to define a new
 3837 restricted datatype. For our example, this is the **qdt:RestrictedCurrencyCodeType**. Although in our
 3838 data model this is a restriction of the **qdt:CurrencyCodeType**, this new datatype's restriction
 3839 declaration will have a base value of **xsd:token** rather than **CurrencyCodeType** because of XSD
 3840 limitations. In XSD, enumerations are repeating facets and the nature of **xsd:restriction** is such that
 3841 the set of facets in a restricted type is the sum of the facets for the original type and the restricted type –
 3842 actually resulting in an extension rather than restriction. For our example, the new **xsd:simpleType**
 3843 definition would occur in a new code list schema module:

```
3844 <xsd:simpleType name="RestrictedCurrencyCodeContentType">
3845   <xsd:restriction base="xsd:token">
3846     <xsd:enumeration value="AED">
3847       <xsd:annotation>
3848         <xsd:documentation>
3849           <CodeName>Dirham</CodeName>
3850         </xsd:documentation>
3851       </xsd:annotation>
3852     </xsd:enumeration>
3853   </xsd:restriction>
3854 </xsd:simpleType>
```

3855 In the instance documents, allowed values of the element **RestrictedCurrencyCode** are limited to those
 3856 contained in the restricted code list schema module.

D.2 The Use of Identifier Schemes within XML Schemas

3857 The UN/CEFACT XML NDR allows for five alternative uses for identifier schemes:

- 3859 • Referencing a predefined standard identifier scheme, such as agency identifiers according to DE
 3860 3055, as a supplementary component in an unqualified data type, such as **udt:codeType**.
- 3861 • Referencing any identifier scheme, standard or proprietary, by providing the required identification
 3862 as attributes in the unqualified data type **udt:IdentifierType**
- 3863 • Referencing a predefined identifier scheme by declaring a specific qualified data type
- 3864 • Choosing or combining values from several identifier schemes
- 3865 • Restricting allowed identifier values

3866 The rules for identifier schemes are the same as those for code lists, thus the examples found in D.1 also
 3867 apply to identifier lists

3868

Appendix E Annotation Templates

The following templates define the annotation for each of the schema modules.

```
<!-- Root Schema Documentation -->
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueId></ccts:UniqueId>
    <ccts:Acronym>RSM</ccts:Acronym>
    <ccts:Name></ccts:Name>
    <ccts:Version></ccts:Version>
    <ccts:Definition></ccts:Definition>
    <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
    <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
    <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
    <ccts:ProductContextValue></ccts:ProductContextValue>
    <ccts:IndustryContextValue></ccts:IndustryContextValue>
    <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
    <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
    <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
  </xsd:documentation>
</xsd:annotation>

<!-- ABIE Documentation -->
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueId></ccts:UniqueId>
    <ccts:Acronym>ABIE</ccts:Acronym>
    <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
    <ccts:Version></ccts:Version>
    <ccts:Definition></ccts:Definition>
    <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
    <ccts:ObjectClassQualifierTerm></ccts:ObjectClassQualifierTerm>
    <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
    <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
    <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
    <ccts:ProductContextValue></ccts:ProductContextValue>
    <ccts:IndustryContextValue></ccts:IndustryContextValue>
    <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
    <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
    <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
    <ccts:UsageRule></ccts:UsageRule>
    <ccts:BusinessTerm></ccts:BusinessTerm>
    <ccts:Example></ccts:Example>
  </xsd:documentation>
</xsd:annotation>

<!-- BBIE Documentation -->
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueId></ccts:UniqueId>
    <ccts:Acronym>ABIE</ccts:Acronym>
    <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
    <ccts:Version></ccts:Version>
    <ccts:Definition></ccts:Definition>
    <ccts:Cardinality></ccts:Cardinality>
    <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
    <ccts:ObjectClassQualifierTerm></ccts:ObjectClassQualifierTerm>
    <ccts:PropertyTerm></ccts:PropertyTerm>
    <ccts:PropertyQualifierTerm></ccts:PropertyQualifierTerm>
```

```

3925 <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
3926 <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3927 <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3928 <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
3929 <ccts:ProductContextValue></ccts:ProductContextValue>
3930 <ccts:IndustryContextValue></ccts:IndustryContextValue>
3931 <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3932 <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3933 <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3934 <ccts:UsageRule></ccts:UsageRule>
3935 <ccts:BusinessTerm></ccts:BusinessTerm>
3936 <ccts:Example></ccts:Example>
3937 </xsd:documentation>
3938 </xsd:annotation>

3939 <!-- ASBIE Documentation -->
3940 <xsd:annotation>
3941     <xsd:documentation xml:lang="en">
3942         <ccts:UniqueID></ccts:UniqueID>
3943         <ccts:Acronym>ABIE</ccts:Acronym>
3944         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3945         <ccts:Version></ccts:Version>
3946         <ccts:Definition></ccts:Definition>
3947         <ccts:Cardinality></ccts:Cardinality>
3948         <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
3949         <ccts:ObjectClassQualifierTerm></ccts:ObjectClassQualifierTerm>
3950         <ccts:PropertyTerm></ccts:PropertyTerm>
3951         <ccts:PropertyQualifierTerm></ccts:PropertyQualifierTerm>
3952         <ccts:AssociatedObjectClassTerm></ccts:AssociatedObjectClassTerm>
3953         <ccts:AssociatedObjectClassQualifierTerm></ccts:AssociatedObjectClassQualifierTerm>
3954         <ccts:AssociationType></ccts:AssociationType>
3955         <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3956         <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3957         <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>
3958         <ccts:ProductContextValue></ccts:ProductContextValue>
3959         <ccts:IndustryContextValue></ccts:IndustryContextValue>
3960         <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3961         <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3962         <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3963         <ccts:UsageRule></ccts:UsageRule>
3964         <ccts:BusinessTerm></ccts:BusinessTerm>
3965         <ccts:Example></ccts:Example>
3966     </xsd:documentation>
3967 </xsd:annotation>

3968 <!-- Qualified Data Type Documentation -->
3969 <xsd:annotation>
3970     <xsd:documentation xml:lang="en">
3971         <ccts:UniqueID></ccts:UniqueID>
3972         <ccts:Acronym>QDT</ccts:Acronym>
3973         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3974         <ccts:Version></ccts:Version>
3975         <ccts:Definition></ccts:Definition>
3976         <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
3977         <ccts:DataTypeQualifierTerm></ccts:DataTypeQualifierTerm>
3978         <ccts:PrimitiveType></ccts:PrimitiveType>
3979         <ccts:BusinessProcessContextValue></ccts:BusinessProcessContextValue>
3980         <ccts:GeopoliticalOrRegionContextValue></ccts:GeopoliticalOrRegionContextValue>
3981         <ccts:OfficialConstraintContextValue></ccts:OfficialConstraintContextValue>

```

```

3982 <ccts:ProductContextValue></ccts:ProductContextValue>
3983 <ccts:IndustryContextValue></ccts:IndustryContextValue>
3984 <ccts:BusinessProcessRoleContextValue></ccts:BusinessProcessRoleContextValue>
3985 <ccts:SupportingRoleContextValue></ccts:SupportingRoleContextValue>
3986 <ccts:SystemCapabilitiesContextValue></ccts:SystemCapabilitiesContextValue>
3987 <ccts:UsageRule></ccts:UsageRule>
3988 <ccts:BusinessTerm></ccts:BusinessTerm>
3989 <ccts:Example></ccts:Example>
3990 </xsd:documentation>
3991 </xsd:annotation>

3992 <!-- Qualified Data Type Supplementary Component Documentation-->
3993 <xsd:annotation>
3994   <xsd:documentation xml:lang="en">
3995     <ccts:UniqueID></ccts:UniqueID>
3996     <ccts:Acronym>SC</ccts:Acronym>
3997     <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3998     <ccts:Definition></ccts:Definition>
3999     <ccts:Cardinality></ccts:Cardinality>
4000     <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
4001     <ccts:PropertyTerm></ccts:PropertyTerm>
4002     <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4003     <ccts:PrimitiveType></ccts:PrimitiveType>
4004     <ccts:UsageRule></ccts:UsageRule>
4005   </xsd:documentation>
4006 </xsd:annotation>

4007 <!-- Unqualified Data Type Documentation-->
4008 <xsd:annotation>
4009   <xsd:documentation xml:lang="en">
4010     <ccts:UniqueID></ccts:UniqueID>
4011     <ccts:Acronym>UDT</ccts:Acronym>
4012     <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4013     <ccts:Version></ccts:Version>
4014     <ccts:Definition></ccts:Definition>
4015     <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4016     <ccts:PrimitiveType></ccts:PrimitiveType>
4017     <ccts:UsageRule></ccts:UsageRule>
4018   </xsd:documentation>
4019 </xsd:annotation>

4020 <!-- Unqualified Data Type Supplementary Component Documentation-->
4021 <xsd:annotation>
4022   <xsd:documentation xml:lang="en">
4023     <ccts:UniqueID></ccts:UniqueID>
4024     <ccts:Acronym>SC</ccts:Acronym>
4025     <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4026     <ccts:Definition></ccts:Definition>
4027     <ccts:Cardinality></ccts:Cardinality>
4028     <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
4029     <ccts:PropertyTerm></ccts:PropertyTerm>
4030     <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4031     <ccts:PrimitiveType></ccts:PrimitiveType>
4032     <ccts:UsageRule></ccts:UsageRule>
4033   </xsd:documentation>
4034 </xsd:annotation>

4035 <!-- Core Component Type Documentation -->
4036 <xsd:annotation>
4037   <xsd:documentation xml:lang="en">

```

```

4038 <ccts:UniqueID></ccts:UniqueID>
4039 <ccts:Acronym>CCT</ccts:Acronym>
4040 <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4041 <ccts:Version></ccts:Version>
4042 <ccts:Definition></ccts:Definition>
4043 <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4044 <ccts:PrimitiveType></ccts:PrimitiveType>
4045 <ccts:UsageRule></ccts:UsageRule>
4046 </xsd:documentation>
4047 </xsd:annotation>
4048 <!-- Core Component Type Supplementary Component Documentation-->
4049 <xsd:annotation>
4050   <xsd:documentation xml:lang="en">
4051     <ccts:UniqueID></ccts:UniqueID>
4052     <ccts:Acronym>SC</ccts:Acronym>
4053     <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
4054     <ccts:Definition></ccts:Definition>
4055     <ccts:Cardinality></ccts:Cardinality>
4056     <ccts:ObjectClassTerm></ccts:ObjectClassTerm>
4057     <ccts:PropertyTerm></ccts:PropertyTerm>
4058     <ccts:PrimaryRepresentationTerm></ccts:PrimaryRepresentationTerm>
4059     <ccts:PrimitiveType></ccts:PrimitiveType>
4060     <ccts:UsageRule></ccts:UsageRule>
4061   </xsd:documentation>
4062 </xsd:annotation>
4063 <!-- Code List / Identification Schema Documentation-->
4064 <xsd:annotation>
4065   <xsd:documentation xml:lang="en">
4066     <ccts:Name></ccts:Name>
4067     <ccts:Description></ccts:Description>
4068   </xsd:documentation>
4069 </xsd:annotation>
4070

```

Appendix F Naming and Design Rules Checklist

- 4071 [R1] Conformance shall be determined through adherence to the content of normative sections, rules and definitions.
- 4072 [R2] All UN/CEFACT XSD Schema design rules MUST be based on the *W3C XML SchemaRecommendations: XML Schema Part 1: Structures and XML Schema Part 2: Data Types*
- 4073 [R3] All UN/CEFACT XSD Schema and UN/CEFACT conformant XML instance documents MUST be based on the W3C suite of technical specifications holding recommendation status.
- 4074 [R4] UN/CEFACT XSD Schema MUST follow the standard structure defined in Appendix B
- 4075 [R5] Each element or attribute XML name MUST have one and only one fully qualified
- 4076 [R6] Element, attribute and type names MUST be composed of words in the English language, using the primary English spellings provided in the Oxford English Dictionary.
- 4077 [R7] Lower camel case (LCC) MUST be used for naming attributes
- 4078 [R8] Upper camel case (UCC) MUST be used for naming elements and types.
- 4079 [R9] Element, attribute and type names MUST be in singular form unless the concept itself is plural.
- 4080 [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.
- 4081 [R11] This rule has been combined with [R10].
- 4082 [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.
- 4083 [R13] The acronyms and abbreviations listed in Appendix C MUST always be used.
- 4084 [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.
- 4085 [R15] Acronyms MUST appear in all upper case for all element declarations and type definitions.
- 4086 [R16] The schema module file name for modules other than code lists or identifier lists MUST be of the form **<SchemaModuleName>_<Version>.xsd**, with periods, spaces, or other separators and the words Schema Module removed.
- 4087 [R17] The schema module file name for code lists and identifier lists, MUST be of the form **<AgencyName>_<ListName>_<Version>.xsd**, with periods, spaces, or other separators removed.
- 4088 [R18] In representing versioning schemes in file names, only the major version should be included.
- 4089 [R19] A root schema MUST be created for each unique business information payload.
- 4090 [R20] Each UN/CEFACT root schema module MUST be named **<BusinessInformationPayload> Schema Module**.
- 4091 [R21] A root schema MUST NOT replicate reusable constructs available in schema modules capable of being referenced through **xsd:include** or **xsd:import**.
- 4092 [R22] UN/CEFACT XSD schema modules MUST either be treated as external schema modules, or as internal schema modules of the root schema.
- 4093 [R23] All UN/CEFACT internal schema modules MUST be in the same namespace as their corresponding **rsm:RootSchema**.
- 4094 [R24] Each UN/CEFACT internal schema module MUST be named **<ParentRootSchemaModuleName><InternalSchemaModuleFunction> Schema Module**
- 4095 [R25] A Core Component Type schema module MUST be created.
- 4096 [R26] The **cct:CoreComponentType** schema module MUST be named 'Core Component Type Schema Module'.
- 4097 [R203] An Unqualified Data Type MUST NOT contain any restriction on their source CCTs other than those defined in CCTS and agreed upon best practices.
- 4098 [R27] An Unqualified Data Type schema module MUST be created
- 4099 [R28] The **udt:UnqualifiedDataType** schema module MUST be named 'Unqualified Data TypeSchema Module'

- 4126 [R29] A Qualified Data Type schema module MUST be created.
- 4127 [R30] The `qdt:QualifiedDataType` schema module MUST be named 'Qualified Data Type Schema Module'.
- 4129 [R31] A Reusable Aggregate Business Information Entity schema module MUST be created.
- 4130 [R32] The `ram:ReusableAggregateBusinessInformationEntity` schema module MUST be named 'Reusable Aggregate Business Information Entity Schema Module'.
- 4132 [R33] Reusable Code List schema modules MUST be created to convey code list enumerations
- 4133 [R34] The name of each `clm:CodeList` schema module MUST be of the form: `<Code List Agency Identifier|Code List Agency Name><Code List Identification Identifier|Code List Name>` - Code List Schema Module
- 4136 Where:
 Code List Agency Identifier = Identifies the agency that maintains the code list
 Code List Agency Name = Agency that maintains the code list
 Code List Identification Identifier = Identifies a list of the respective corresponding codes
 Code List Name = The name of the code list as assigned by the agency that maintains the code list
- 4142 [R35] An identifier list schema module MUST be created to convey enumerated values for each identifier list that requires runtime validation.
- 4144 [R36] The name of each `ids:IdentifierList` schema module MUST be of the form:
`<Identifier Scheme Agency Identifier|Identifier Scheme Agency Name><Identifier Scheme Identifier|Identifier Scheme Name>` - Identifier List Schema Module
- 4148 Where:
 Identifier Scheme Agency Identifier = The identification of the agency that maintains the identifier list
 Identifier Scheme Agency Name = Agency that maintains the identifier list
 Identifier Scheme Identifier = The identification of the identifier list
 Identifier Scheme Name = Name as assigned by the agency that maintains the identifier list
- 4154 [R37] Imported schema modules MUST be fully conformant with the UN/CEFACT XML Naming and Design Rules Technical Specification and the UN/CEFACT Core Components Technical Specification.
- 4157 [R38] Every UN/CEFACT defined or imported schema module MUST have a namespace declared, using the `xsd:targetNamespace` attribute.
- 4159 [R39] Every version of a defined or imported schema module other than internal schema modules MUST have its own unique namespace.
- 4161 [R40] UN/CEFACT published namespace declarations MUST NOT be changed, and its contents MUST NOT be changed unless such change does not break backward compatibility.
- 4163 [R41] UN/CEFACT namespaces MUST be defined as Uniform Resource Names.
- 4164 [R42] The names for namespaces MUST have the following structure while the schema is at draft status:
`urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>`
- 4167 Where:
 schematype = a token identifying the type of schema module:
`data|process|codelist|identifierlist|documentation`
 status = a token identifying the standards status of the schema module: `draft|standard`
 name = the name of the schema module (using upper camel case) with periods, spaces, or other separators and the words 'schema module' removed.
 major = the major version number. Sequentially assigned, first release starting with the number 1.
- 4174 [R43] This rule was combined with [R42].
- 4175 [R44] UN/CEFACT namespace values will only be assigned to UN/CEFACT developed objects.
- 4176 [R45] The general structure for schema location MUST be:
`./<schematype>/<status>/<name>_<major>. <minor>[p <revision>].xsd`
- 4178 Where:
 schematype = a token identifying the type of schema module:
`data|process|codelist|identifierlist|documentation`
 status = the status of the schema as: `draft|standard`

- 4182 name = the name of the schema module (using upper camel case) with periods, spaces, or
 4183 other separators and the words 'schema module' removed.
 4184 major = the major version number, sequentially assigned, first release starting with the number 1.
 4185 minor = the minor version number within a major release, sequentially assigned, first release
 4186 starting with the number 0.
 4187 revision = sequentially assigned alphanumeric character for each revision of a minor release. Only
 4188 applicable where status = draft.
 4189 [R46] Each **xsd:schemaLocation** attribute declaration MUST contain a resolvable URL, and in the
 4190 case of an absolute path, a persistent URL.
 4191 [R47] This rule has been removed.
 4192 [R48] The **xsd:schema** version attribute MUST always be declared.
 4193 [R49] The **xsd:schema** version attribute MUST use the following template:
 4194 **<xsd:schema ... version=".">**
 4195 [R50] Every schema version namespace declaration MUST have the URI of:
 4196 **urn:un:unece:uncefact:<schemaType>:<status>:<name>:<major>**
 4197 [R51] Every UN/CEFACT XSD Schema and schema module major version number MUST be a
 4198 sequentially assigned incremental integer greater than zero.
 4199 [R52] Minor versioning MUST be limited to declaring new optional XSD constructs, extending
 4200 existing XSD constructs, or refinements of an optional nature.
 4201 [R53] For UN/CEFACT minor version changes, the name of the schema construct MUST NOT change.
 4202 [R54] Changes in minor versions MUST NOT break semantic compatibility with prior versions
 4203 having the same major version number.
 4204 [R55] UN/CEFACT minor version schema MUST incorporate all XML constructs from the
 4205 immediately preceding major or minor version schema.
 4206 [R56] The **xsd:elementFormDefault** attribute MUST be declared and its value set to **qualified**.
 4207 [R57] The **xsd:attributeFormDefault** attribute MUST be declared and its value set to
 4208 **unqualified**.
 4209 [R58] The **xsd** prefix MUST be used in all cases when referring to <http://www.w3.org/2001/XMLSchema> as follows:
 4210 **xmlns:xsd=http://www.w3.org/2001/XMLSchema**.
 4211 [R59] **xsd:appInfo** MUST NOT be used.
 4212 [R60] **xsd:notation** MUST NOT be used.
 4213 [R61] **xsd:wildcard** MUST NOT be used.
 4214 [R62] The **xsd:any** element MUST NOT be used.
 4215 [R63] The **xsd:any** attribute MUST NOT be used.
 4216 [R64] Mixed content MUST NOT be used (excluding documentation).
 4217 [R65] **xsd:substitutionGroup** MUST NOT be used.
 4218 [R66] **xsd:ID/xsd:IDREF** MUST NOT be used.
 4219 [R67] **xsd:key/xsd:keyref** MUST be used for information association.
 4220 [R68] The absence of a construct or data MUST NOT carry meaning.
 4221 [R69] User declared attributes MUST only be used to convey core component type (CCT)
 4222 supplementary component information.
 4223 [R70] A **xsd:attribute** that represents a supplementary component with variable information MUST
 4224 be based on the appropriate XSD built-in data type.
 4225 [R71] A **xsd:attribute** that represents a supplementary component which represents codes MUST
 4226 be based on the **xsd:simpleType** of the appropriate code list.
 4227 [R72] A **xsd:attribute** that represents a supplementary component which represents identifiers
 4228 MUST be based on the **xsd:simpleType** of the appropriate identifier scheme.
 4229 [R73] The **xsd:nillable** attribute MUST NOT be used.
 4230 [R74] Empty elements MUST NOT be used.
 4231 [R75] Every BBIE leaf element declaration MUST be of the **udt:UnqualifiedDataType** or
 4232 **qdt:QualifiedDataType** that represents the source basic business information entity
 4233 (BBIE) data type.
 4234 [R76] The **xsd:all** element MUST NOT be used.
 4235 [R77] All type definitions MUST be named.

- 4237 [R78] Data type definitions with the same semantic meaning MUST NOT have an identical set of
4238 facet restrictions.
- 4239 [R79] **xsd:extension** MUST only be used in the **cct:CoreComponentType** schema module and
4240 the **udt:UnqualifiedDataType** schema module. When used it MUST only be used for
4241 declaring **xsd:attributes** to accommodate relevant supplementary components.
- 4242 [R80] When **xsd:restriction** is applied to a **xsd:simpleType** or **xsd:complexType** that
4243 represents a data type the derived construct MUST use a different name.
- 4244 [R81] Each UN/CEFACT defined or declared construct MUST use the **xsd:annotation** element for
4245 required CCTS documentation.
- 4246 [R82] The root schema module MUST be represented by a unique token.
- 4247 [R83] The **rsm:RootSchema** MUST import the following schema modules:
– **ram:ReusableABIE** Schema Module
– **udt:UnqualifiedDataType** Schema Module
– **qdt:QualifiedDataType** Schema Module
- 4248 [R84] A **rsm:RootSchema** in one UN/CEFACT namespace that is dependent upon type definitions or
4249 element declaration defined in another namespace MUST import the **rsm:RootSchema** from
4250 that namespace.
- 4251 [R85] A **rsm:RootSchema** in one UN/CEFACT namespace that is dependent upon type definitions or
4252 element declarations defined in another namespace MUST NOT import Schema Modules from
4253 that namespace other than the **rsm:RootSchema**.
- 4254 [R86] The **rsm:RootSchema** MUST include any internal schema modules that reside in the root
4255 schema namespace.
- 4256 [R87] A single global element known as the root element, representing the business information
4257 payload, MUST be declared in a **rsm:RootSchema**.
- 4258 [R88] The name of the root element MUST be the name of the business information payload with
4259 separators and spaces removed.
- 4260 [R89] The root element declaration must be of **xsd:complexType** that represents the business
4261 information payload.
- 4262 [R90] Root schema MUST define a single **xsd:complexType** that fully describes the business
4263 information payload.
- 4264 [R91] The name of the root schema **xsd:complexType** MUST be the name of the root element with
4265 the word 'Type' appended.
- 4266 [R92] The **rsm:RootSchema** root element declaration MUST have a structured set of annotations
4267 present in the following pattern:
○ UniqueID (mandatory): The identifier that references the business information payload
4268 instance in a unique and unambiguous way.
○ Acronym (mandatory): The abbreviation of the type of component. In this case the value will
4269 always be RSM.
○ Name (mandatory): The name of the business information payload.
○ Version (mandatory): An indication of the evolution over time of a business information
4270 payload.
○ Definition (mandatory): A brief description of the business information payload.
○ BusinessProcessContextValue (mandatory, repetitive): The business process with which this
4271 business information is associated.
○ GeopoliticalRegionContextValue (optional, repetitive): The geopolitical/region contexts for
4272 this business information payload.
○ OfficialConstraintContextValue (optional, repetitive): The official constraint context for this
4273 business information payload.
○ ProductContextValue (optional, repetitive): The product context for this business information
4274 payload.
○ IndustryContextValue (optional, repetitive): The industry context for this business information
4275 payload.
○ BusinessProcessRoleContextValue (optional, repetitive): The role context for this business
4276 information payload.
○ SupportingRoleContextValue (optional, repetitive): The supporting role context for this
4277 business information payload.

- SystemCapabilitiesContextValue (optional, repetitive): The system capabilities context for this business information payload.
- [R93] All UN/CEFACT internal schema modules MUST be in the same namespace as their corresponding **rsm:RootSchema**.
- [R94] The internal schema module MUST be represented by the same token as its **rsm:RootSchema**.
- [R95] The Reusable Aggregate Business Information Entity schema module MUST be represented by the token **ram**.
- [R96] The **ram:ReusableAggregateBusinessInformationEntity** schema MUST import the following schema modules:
 - **udt:UnqualifiedDataType** Schema Module
 - **qdt:QualifiedDataType** Schema Module
- [R97] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named **xsd:complexType** MUST be defined.
- [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’.
- [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.
- [R100] Recursion of **xsd:sequence** and/or **xsd:choice** MUST NOT occur.
- [R101] The order and cardinality of the elements within an ABIE **xsd:complexType** MUST be according to the structure of the ABIE as defined in the model.
- [R102] For each ABIE, a named **xsd:element** MUST be globally declared.
- [R103] The name of the ABIE **xsd:element** MUST be the **ccts:DictionaryEntryName** with the separators and ‘Details’ suffix removed and approved abbreviations and acronyms applied.
- [R104] Every ABIE global element declaration MUST be of the **xsd:complexType** that represents the ABIE.
- [R105] For every BBIE identified in an ABIE, a named **xsd:element** MUST be locally declared within the **xsd:complexType** representing that ABIE.
- [R106] Each BBIE element name declaration MUST be the property term and qualifiers and the representation term of the basic business information entity (BBIE). Where the word ‘identification’ is the final word of the property term and the representation term is ‘identifier’, the term ‘identification’ MUST be removed. Where the word ‘indication’ is the final word of the property term and the representation term is ‘indicator’, the term ‘indication’ MUST be removed from the property term.
- [R107] If the representation term of a BBIE is ‘text’, ‘text’ MUST be removed.
- [R108] The BBIE element MUST be based on an appropriate data type that is defined in the UN/CEFACT **qdt:QualifiedDataType** or **udt:UnqualifiedDataType** schema modules.
- [R109] For every ASBIE whose **ccts:AssociationType** is a composition, a named **xsd:element** MUST be locally declared.
- [R110] For each locally declared ASBIE, the element name MUST be the ASBIE property term and qualifier term(s) and the object class term and qualifier term(s) of the associated ABIE.
- [R111] For each locally declared ASBIE, the element declaration MUST be of the **sd:complexType** that represents its associated ABIE.
- [R112] For every ASBIE whose **ccts:AssociationType** is not a composition, the globally declared element for the associated ABIE must be referenced using **xsd:ref**.
- [R113] For every ABIE **xsd:complexType** and **xsd:element** definition a structured set of annotations MUST be present in the following pattern:
 - UniqueID (mandatory): The identifier that references an ABIE instance in a unique and unambiguous way.
 - Acronym (mandatory): The abbreviation of the type of component. In this case the value will always be ABIE.
 - DictionaryEntryName (mandatory): The official name of an ABIE.
 - Version (mandatory): An indication of the evolution over time of an ABIE instance.

- Definition (mandatory): The semantic meaning of an ABIE.
 - ObjectClassTerm (mandatory): 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.
 - GeopoliticalRegionContextValue (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].
- [R115] For every BBIE **xsd:element** declaration a structured set of annotations MUST be present in the following pattern:
- UniqueID (mandatory): The identifier that references a BBIE instance in a unique and unambiguous way.
 - Acronym (mandatory): The abbreviation of the type of component. In this case the value will always be BBIE.
 - DictionaryEntryName (mandatory): The official name of the BBIE.
 - VersionID (mandatory): An indication of the evolution over time of a BBIE instance.
 - Definition (mandatory): The semantic meaning of the BBIE.
 - Cardinality (mandatory): Indication whether the BIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.
 - ObjectClassTerm (mandatory): The Object Class Term of the parent ABIE.
 - ObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the parent ABIE.
 - PropertyTerm (mandatory): The Property Term of the BBIE.
 - PropertyQualifierTerm (optional): Qualifies the Property Term of the BBIE.
 - PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the BBIE.
 - BusinessProcessContextValue (optional, repetitive): The business process with which this BBIE is associated.
 - GeopoliticalRegionContextValue (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.
- [R116] For every ASBIE **xsd:element** declaration a structured set of annotations MUST be present in the following pattern:
- UniqueID (mandatory): The identifier that references an ASBIE instance in a unique and unambiguous way.
 - Acronym (mandatory): The abbreviation of the type of component. In this case the value will always be ASBIE.
 - DictionaryEntryName (mandatory): The official name of the ASBIE.
 - Version (mandatory): An indication of the evolution over time of the ASBIE instance.
 - Definition (mandatory): The semantic meaning of the ASBIE.
 - Cardinality (mandatory): Indication whether the ASBIE Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the ABIE.
 - ObjectClassTerm (mandatory): The Object Class Term of the associating ABIE.
 - ObjectClassQualifierTerm (optional): A term that qualifies the Object Class Term of the associating ABIE.
 - PropertyTerm (mandatory): The Property Term of the ASBIE.
 - PropertyQualifierTerm (Optional): A term that qualifies the Property Term of the ASBIE.
 - AssociatedObjectClassTerm (mandatory): The Object Class Term of the associated ABIE.
 - AssociatedObjectClassQualifierTerm (optional): Qualifies the Object Class Term of the associated ABIE.
 - AssociationType (mandatory): The Association Type of the ASBIE.
 - BusinessProcessContextValue (optional, repetitive): The business process with which this ASBIE is associated.
 - GeopoliticalRegionContextValue (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.
 - BusinessTerm (optional, repetitive): A synonym term under which the ASBIE is commonly known and used in the business.
 - Example (optional, repetitive): Example of a possible value of an ASBIE.
- [R117] The core component type (CCT) schema module MUST be represented by the token cct.
- [R118] The **cct:CoreCoreComponentType** schema module MUST NOT include or import any other schema modules.
- [R119] Every core component type MUST be defined as a named **xsd:complexType** in the **cct:CoreComponentType** schema module.
- [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.

- 4448 [R121] Each core component type **xsd:complexType** definition MUST contain one
 4449 **xsd:simpleContent** element.
- 4450 [R122] The core component type **xsd:complexType** definition **xsd:simpleContent** element MUST
 4451 contain one **xsd:extension** element. This **xsd:extension** element must include an XSD
 4452 based attribute that defines the specific XSD built-in data type required for the CCT content
 4453 component.
- 4454 [R123] Within the core component type **xsd:extension** element a **xsd:attribute** MUST be
 4455 declared for each supplementary component pertaining to that core component type.
- 4456 [R124] Each core component type supplementary component **xsd:attribute** name MUST be the
 4457 CCTS supplementary component dictionary entry name with the separators and spaces
 4458 removed.
- 4459 [R125] If the object class of the supplementary component dictionary entry name contains the name of
 4460 the representation term of the parent CCT, the duplicated object class word or words MUST be
 4461 removed from the supplementary component **xsd:attribute** name.
- 4462 [R126] If the object class of the supplementary component dictionary entry name contains the term
 4463 ‘identification’, the term ‘identification’ MUST be removed from the supplementary component
 4464 **xsd:attribute** name.
- 4465 [R127] If the representation term of the supplementary component dictionary entry name is ‘text’, the
 4466 representation term MUST be removed from the supplementary component **xsd:attribute**
 4467 name.
- 4468 [R128] The attribute representing the supplementary component MUST be based on the appropriate XSD
 4469 built-in data type.
- 4470 [R129] For every core component type **xsd:complexType** definition a structured set of annotations
 4471 MUST be present in the following pattern:
- 4472 ○ UniqueID (mandatory): The identifier that references the Core Component Type instance in a
 4473 unique and unambiguous way.
 - 4474 ○ Acronym (mandatory): The abbreviation of the type of component. . In this case the value will
 4475 always be CCT.
 - 4476 ○ DictionaryEntryName (mandatory): The official name of a Core Component Type.
 - 4477 ○ Version (mandatory): An indication of the evolution over time of a Core Component Type
 4478 instance.
 - 4479 ○ Definition (mandatory): The semantic meaning of a Core Component Type.
 - 4480 ○ PrimaryRepresentationTerm (mandatory): The primary representation term of the Core
 4481 Component Type.
 - 4482 ○ PrimitiveType (mandatory): The primitive data type of the Core Component Type.
 - 4483 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 4484 applicable to the Core Component Type.
- 4485 [R130] For every supplementary component **xsd:attribute** declaration a structured set of
 4486 annotations MUST be present in the following pattern:
- 4487 ○ UniqueID (optional): The identifier that references the Supplementary Component instance in
 4488 a unique and unambiguous way.
 - 4489 ○ Acronym (mandatory): The abbreviation of the type of Supplementary Component. In this
 4490 case the value will always be SC.
 - 4491 ○ DictionaryEntryName (mandatory): The official name of the Supplementary Component.
 - 4492 ○ Definition (mandatory): The semantic meaning of the Supplementary Component.
 - 4493 ○ Cardinality (mandatory): The cardinality of the Supplementary Component.
 - 4494 ○ ObjectClassTerm (mandatory): The Object Class of the Supplementary Component.
 - 4495 ○ PropertyTerm (mandatory): The Property Term of the Supplementary Component.
 - 4496 ○ PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the
 4497 Supplementary Component.
 - 4498 ○ PrimitiveType (mandatory): The primitive data type of the Supplementary Component.
 - 4499 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 4500 applicable to the Supplementary Core Component.
- 4501 [R131] The Unqualified Data Type schema module namespace MUST be represented by the token **udt**.
- 4502 [R132] The **udt:UnqualifiedDataType** schema MUST only import the following schema
 4503 modules: – **ids:IdentifierList** schema modules – **clm:CodeList** schema modules

- 4504 [R133] An unqualified data type MUST be defined for each approved primary and secondary
4505 representation terms identified in the CCTS Permissible Representation Terms table.
4506 [R134] The name of each unqualified data type MUST be the dictionary entry name of the primary or
4507 secondary representation term, with the word 'Type' appended, the separators and spaces
4508 removed and approved abbreviations applied.
4509 [R135] For every unqualified data type whose supplementary components map directly to the properties
4510 of a XSD built-in data type, the unqualified data type MUST be defined as a named
4511 **xsd:simpleType** in the **udt:UnqualifiedDataType** schema module.
4512 [R136] Every unqualified data type **xsd:simpleType** MUST contain one **xsd:restriction**
4513 element. This **xsd:restriction** element MUST include an **xsd:base** attribute that
4514 defines the specific XSD built-in data type required for the content component.
4515 [R137] For every unqualified data type whose supplementary components are not equivalent to the
4516 properties of a XSD built-in data type, the unqualified data type MUST be defined as an
4517 **xsd:complexType** in the **udt:UnqualifiedDataType** schema module.
4518 [R138] Every unqualified data type **xsd:complexType** definition MUST contain one
4519 **xsd:simpleContent** element.
4520 [R139] Every unqualified data type **xsd:complexType xsd:simpleContent** element MUST
4521 contain one **xsd:extension** element. This **xsd:extension** element must include an
4522 **xsd:base** attribute that defines the specific XSD built-in data type required for the content
4523 component.
4524 [R204] When a combination of the complex and simple types are necessary to support business
4525 requirements, the element MUST be declared as an **xsd:complexType** with an
4526 **xsd:choice** between elements declared as the two different alternatives.
4527 [R140] Within the unqualified data type **xsd:complexType xsd:extension** element an
4528 **xsd:attribute** MUST be declared for each supplementary component pertaining to the
4529 underlying CCT.
4530 [R141] Each supplementary component **xsd:attribute** name MUST be the supplementary
4531 component name with the separators and spaces removed, and approved abbreviations and
4532 acronyms applied.
4533 [R142] If the object class of the supplementary component dictionary entry name contains the name of
4534 the representation term, the duplicated object class word or words MUST be removed from the
4535 supplementary component **xsd:attribute** name.
4536 [R143] If the object class of the supplementary component dictionary entry name contains the term
4537 'identification', the term 'identification' MUST be removed from the supplementary component
4538 **xsd:attribute** name.
4539 [R144] If the representation term of the supplementary component dictionary entry name is 'text', the
4540 representation term MUST be removed from the supplementary component **xsd:attribute**
4541 name.
4542 [R145] If the representation term of the supplementary component is 'Code' and validation is required,
4543 then the attribute representing this supplementary component MUST be based on the defined
4544 **xsd:simpleType** of the appropriate external imported code list.
4545 [R146] If the representation term of the supplementary component is 'Identifier' and validation is
4546 required, then the attribute representing this supplementary component MUST be based on the
4547 defined **xsd:simpleType** of the appropriate external imported identifier list.
4548 [R147] If the representation term of the supplementary component is other than 'Code' or 'Identifier',
4549 then the attribute representing this supplementary component MUST be based on the
4550 appropriate XSD built-in data type.
4551 [R148] For every unqualified data type **xsd:complexType** or **xsd:simpleType** definition a
4552 structured set of annotations MUST be present in the following pattern:
4553
 - o UniqueID (mandatory): The identifier that references an Unqualified Data Type instance in a
4554 unique and unambiguous way.
 - o Acronym (mandatory): The abbreviation of the type of component. In this case the value will
4556 always be UDT.
 - o DictionaryEntryName (mandatory): The official name of the Unqualified Data Type.
 - o Version (mandatory): An indication of the evolution over time of the Unqualified Data Type
4559 instance.

- Definition (mandatory): The semantic meaning of the Unqualified Data Type.
- PrimaryRepresentationTerm (mandatory): The primary representation term of the Unqualified Data Type.
- PrimitiveType (mandatory): The primitive data type of the Unqualified Data Type.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the Unqualified Data Type.

- 4560
4561 [R149] For every supplementary component **xsd:attribute** declaration a structured set of
4562 annotations MUST be present in the following pattern:
4563 ○ UniqueID (optional): The identifier that references a Supplementary Component instance in
4564 a unique and unambiguous way.
4565 ○ Acronym (mandatory): The abbreviation of the type of component. In this case the value will
4566 always be SC.
4567 ○ Dictionary Entry Name (mandatory): The official name of the Supplementary Component.
4568 ○ Definition (mandatory): The semantic meaning of the Supplementary Component.
4569 ○ Cardinality (mandatory): The cardinality of the Supplementary Component.
4570 ○ ObjectClassTerm (mandatory): The Object Class of the Supplementary Component.
4571 ○ PropertyTerm (mandatory): The Property Term of the Supplementary Component.
4572 ○ PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the
4573 Supplementary Component.
4574 ○ UsageRule (optional, repetitive): A constraint that describes specific conditions that are
4575 applicable to the Supplementary Component.
- 4576 [R150] The Qualified Data Type schema module namespace MUST be represented by the token **qdt**.
4577 [R151] The **qdt:QualifiedDataType** schema module MUST import the
4578 **udt:UnqualifiedDataType schema module**.
- 4579 [R205] The qdt:QualifiedDataType schema module MUST import all code list and identifier scheme
4580 schemas used in the module.
- 4581 [R152] Where required to change facets of an existing unqualified data type, a new data type MUST be
4582 defined in the **qdt:QualifiedDataType** schema module.
- 4583 [R153] A qualified data type MUST be based on an unqualified or qualified data type and add some
4584 semantic and/or technical restriction to the base data type.
- 4585 [R154] The name of a qualified data type MUST be the name of its base unqualified or qualified data
4586 type with separators and spaces removed and with its qualifier term added.
- 4587 [R155] When a qualified data type is based on an unqualified data type that contains an
4588 **xsd:choice** element, then the qualified data type MUST be based on one or the other of the
4589 elements, but not both.
- 4590 [R156] Every qualified data type based on an unqualified or qualified data type **xsd:complexType**
4591 whose supplementary components do not map directly to the properties of a XSD built-in data
4592 type
4593 MUST be defined as a **xsd:complexType**
4594 MUST contain one **xsd:simpleContent** element
4595 MUST contain one **xsd:restriction** element
4596 MUST include the unqualified data type as its **xsd:base** attribute.
- 4597 [R157] Every qualified data type based on an unqualified or qualified data type
4598 **xsd:simpleType**
4599 MUST contain one **xsd:restriction** element
4600 MUST include the unqualified data type as its **xsd:base** attribute or if the facet
4601 restrictions can be achieved by use of a XSD built-in data type, then that XSD built-in
4602 data type may be used as the **xsd:base** attribute.
- 4603 [R158] Every qualified data type based on a single codelist or identifier list **xsd:simpleType** MUST
4604 contain one **xsd:restriction** element or **xsd:union** element. When using the
4605 **xsd:restriction** element, the **xsd:base** attribute MUST be set to the code list or identifier list
4606 schema module defined simple type with appropriate namespace qualification. When using the
4607 **xsd:union** element, the **xsd:member** type attribute MUST be set to the code list or identifier list
4608 schema module defined simple types with appropriate namespace qualification.
- 4609 [R159] Every qualified data type that has a choice of two or more code lists or identifier lists
4610 MUST be defined as an **xsd:complexType**

- 4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
- MUST contain the **xsd:choice** element whose content model must consist of element references for the alternative code lists or identifier lists to be included with appropriate namespace qualification.
- [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.
- [R161] Every qualified data type definition MUST contain a structured set of annotations in the following sequence and pattern:
- UniqueID (mandatory): The identifier that references a Qualified Data Type instance in a unique and unambiguous way.
 - Acronym (mandatory): The abbreviation of the type of component. In this case the value will always be QDT.
 - DictionaryEntryName (mandatory): The official name of the Qualified Data Type.
 - Version (mandatory): An indication of the evolution over time of the Qualified Data Type instance.
 - Definition (mandatory): The semantic meaning of the Qualified Data Type.
 - PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the Qualified Data Type.
 - DataTypeQualifierTerm (mandatory): A term that qualifies the Representation Term in order to differentiate it from its underlying Unqualified Data Type and other Qualified Data Type.
 - PrimitiveType (mandatory): 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.
- [R162] For every supplementary component **xsd:attribute** declaration a structured set of annotations MUST 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 (mandatory): The abbreviation of the type of component. In this case the value will always be SC.
 - DictionaryEntryName (mandatory): The official name of a Supplementary Component.
 - Definition (mandatory): The semantic meaning of a Supplementary Component.
 - Cardinality (mandatory): Indication whether the Supplementary Component Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Core Component Type.
 - ObjectClassTerm (mandatory): The Object Class Term of the associated Supplementary Component.
 - PropertyTerm (mandatory): The Property Term of the associated Supplementary Component.
 - PrimaryRepresentationTerm (mandatory): The Primary Representation Term of the associated Supplementary Component.
 - PrimitiveType (mandatory): The Primitive Type of the associated Supplementary Component.
 - UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the Supplementary Component.
- [R163] Each UN/CEFACT maintained code list MUST be defined in its own schema module.

- 4674 [R164] Internal code list schema MUST NOT duplicate existing external code list schema when the
 4675 existing ones are available to be imported.
- 4676 [R165] The namespace names for code list schemas MUST have the following structure:
 4677 `urn:un:unece:uncefact:codelist:<status>:<Code List Agency`
 4678 `Identifier|Code List Agency Name Text>:<Code List Identification.`
 4679 `Identifier|Code List Name Text>:<Code List Version. Identifier>`
 4680 Where:
 4681 codelist = this token identifying the schema as a code list
 4682 status = a token identifying the standards status of this code list: draft | standard
 4683 Code List Agency Identifier = identifies the agency that manages a code list. The default
 4684 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.
 4685 Code List Agency Name Text = the name of the agency that maintains the code list.
 4686 Code List Identification Identifier = identifies a list of the respective corresponding codes. listID
 4687 is only unique within the agency that manages this code list. Code List Name Text = the
 4688 name of a list of codes.
 4689 Code List Version Identifier = identifies the version of a code list.
 4690 [R166] This rule was combined with [R165].
 4691 [R167] Each UN/CEFACT maintained code list schema module MUST be represented by a unique
 4692 token constructed as follows:
 4693 `clm[Qualified data type name]<Code List Agency Identifier|Code List`
 4694 `Agency Name Text><Code List Identification Identifier|Code List Name`
 4695 `Text>`
 4696 with any repeated words eliminated.
 4697 [R168] The structure for schema location of code lists MUST be:
 4698 `.../codelist/<status>/<Code List. Agency Identifier|Code List Agency`
 4699 `Name Text>/<Code List Identification Identifier|Code List Name`
 4700 `Text>_<Code List Version Identifier>.xsd`
 4701 Where:
 4702 schematype = a token identifying the type of schema module: codelist
 4703 status = the status of the schema as: draft | standard
 4704 Code List Agency Identifier = identifies the agency that manages a code list. The default
 4705 agencies used are those from DE 3055. Code List Agency Name Text = the name of the
 4706 agency that maintains the code list.
 4707 Code List Identification Identifier = identifies a list of the respective corresponding codes.
 4708 listID is only unique within the agency that manages this code list.
 4709 Code List Name Text = the name of a list of codes.
 4710 Code List Version Identifier = identifies the version of a code list.
 4711 [R169] Each `xsd:schemaLocation` attribute declaration of a code list MUST contain a resolvable
 4712 URL, and if an absolute path is used, it MUST also be persistent.
 4713 [R170] This rule has been removed.
 4714 [R171] Code List schema modules MUST not import or include any other schema modules.
 4715 [R172] Within each code list module one, and only one, named `xsd:simpleType` MUST be defined for
 4716 the content component.
 4717 [R173] The name of the `xsd:simpleType` MUST be the name of code list root element with the
 4718 word 'ContentType' appended.
 4719 [R174] The `xsd:restriction` element base attribute value MUST be set to `xsd:token`.
 4720 [R175] Each code in the code list MUST be expressed as an `xsd:enumeration`, where the
 4721 `xsd:value` for the enumeration is the actual code value.
 4722 [R176] For each code list a single root element MUST be globally declared.
 4723 [R177] The name of the code list root element MUST be the name of the code list following the
 4724 naming rules as defined in section 5.3.
 4725 [R178] The code list root element MUST be of a type representing the actual list of code values.
 4726 [R179] Each code list `xsd:enumeration` MUST contain a structured set of annotations in the
 4727 following sequence and pattern:
 4728
 - Name (mandatory): The name of the code.
 - Description (optional): Descriptive information concerning the code.

- 4730 [R180] Internal identifier lists schema MUST NOT duplicate existing external identifier list schema
 4731 when the existing ones are available to be imported.
- 4732 [R181] Each UN/CEFACT maintained identifier list MUST be defined in its own schema module.
- 4733 [R182] The names for namespaces MUST have the following structure:
 4734 urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme.
 4735 Agency Identifier|Identifier Scheme Agency Name Text>:<Identifier
 4736 Scheme Identifier|Identifier Scheme Name Text>:<Identifier Scheme
 4737 Version Identifier>
 4738 Where:
 4739 status = the token identifying the publication status of this identifier scheme schema =
 4740 draft|standard
 4741 identifierlist = this token identifying the schema as an identifier scheme
 4742 Identifier Scheme Agency Identifier = the identification of the agency that maintains the
 4743 identification scheme.
 4744 Identifier Scheme Agency Name. Text = the name of the agency that maintains the
 4745 identification list.
 4746 Identifier Scheme Identifier = the identification of the identification scheme.
 4747 Identifier Scheme Name. Text = the name of the identification scheme.
 4748 Identifier Scheme Version. Identifier = the version of the identification scheme.
- 4749 [R183] This rule was combined with [R182].
- 4750 [R184] Each UN/CEFACT maintained identifier list schema module MUST be represented by a
 4751 unique token constructed as follows:
 4752 ids[Qualified data type name]<Identification Scheme Agency
 4753 Identifier><Identification Scheme Identifier>
 4754 with any repeated words eliminated.
- 4755 [R185] The structure for schema location of identifier lists MUST be:
 4756 [../identifierlist/<status>/<Identifier Scheme Agency Identifier|Identifier
 4757 Scheme Agency Name Text>/< Identifier Scheme Identifier|Identifier
 4758 Scheme Name Text>_< Identifier Scheme Version Identifier>.xsd
 4759 Where:
 4760 schematype = a token identifying the type of schema module: identifierlist
 4761 status = the status of the schema as: draft|standard
 4762 Identifier Scheme. Agency Identifier = the identification of the agency that maintains the
 4763 identification scheme.
 4764 Identifier Scheme. Agency Name. Text = the name of the agency that maintains the
 4765 identification scheme.
 4766 Identifier Scheme. Identifier = the identification of the identification scheme.
 4767 Identifier Scheme. Name. Text = the name of the identification scheme.
 4768 Identifier Scheme. Version. Identifier = the version of the identification scheme.
- 4769 [R186] Each **xsd:schemaLocation** attribute declaration of an identifier list schema MUST contain a
 4770 resolvable URL, and if an absolute path is used, it MUST also be persistent.
- 4771 [R187] This rule has been removed.
- 4772 [R188] Identifier list schema modules MUST NOT import or include any other schema modules.
- 4773 [R189] Within each identifier list schema module one, and only one, named **xsd:simpleType** MUST
 4774 be defined for the content component.
- 4775 [R190] The name of the **xsd:simpleType** MUST be the name of the identifier list root element
 4776 with the word 'ContentType' appended.
- 4777 [R191] The **xsd:restriction** element base attribute value MUST be set to **xsd:token**.
- 4778 [R192] Each identifier in the identifier list MUST be expressed as an **xsd:enumeration**, where the
 4779 **xsd:value** for the enumeration is the actual identifier value.
- 4780 [R193] Facets other than **xsd:enumeration** MUST NOT be used in the identifier list schema
 4781 module.
- 4782 [R194] For each identifier list a single root element MUST be globally declared.
- 4783 [R195] The name of the identifier list root element MUST be the name of the identifier list following the
 4784 naming rules as defined in section 5.3.

- 4785 [R196] The identifier list root element MUST be of a type representing the actual list of identifier
4786 values.
4787 [R197] Each **xsd:enumeration** MUST contain a structured set of annotations in the following
4788 sequence and pattern:
4789 ○ Name (mandatory): The name of the identifier.
4790 ○ Description (optional): Descriptive information concerning the identifier.
4791 [R198] All UN/CEFACT XML MUST be instantiated using UTF. UTF-8 should be used as the
4792 preferred encoding. If UTF-8 is not used, UTF-16 MUST be used.
4793 [R199] The **xsi** prefix MUST be used where appropriate for referencing **xsd:schemaLocation** and
4794 **xsd:noNamespaceLocation** attributes in instance documents.
4795 [R200] UN/CEFACT conformant instance documents MUST NOT contain an element devoid of
4796 content.
4797 [R201] The **xsi:nil** attribute MUST NOT appear in any conforming instance.
4798 [R202] The **xsi:type** attribute MUST NOT be used

4799

4800 Appendix G: Glossary

- 4801 **Aggregate Business Information Entity (ABIE)** – A collection of related pieces of business information that
4802 together convey a distinct business meaning in a specific *Business Context*. Expressed in modelling terms,
4803 it is the representation of an *Object Class*, in a specific *Business Context*.
- 4804 **Aggregate Core Component - (ACC)** – A collection of related pieces of business information that together
4805 convey a distinct business meaning, independent of any specific *Business Context*. Expressed in modelling
4806 terms, it is the representation of an *Object Class*, independent of any specific *Business Context*.
- 4807 **Aggregation** – An *Aggregation* is a special form of *Association* that specifies a whole-part relationship
4808 between the aggregate (whole) and a component part.
- 4809 **Association Business Information Entity (ASBIE)** - A *Business Information Entity* that represents a complex
4810 business characteristic of a specific *Object Class* in a specific *Business Context*. It has a unique *Business*
4811 *Semantic* definition. An *Association Business Information Entity* represents an *Association Business*
4812 *Information Entity Property* and is therefore associated to an *Aggregate Business Information Entity*, which
4813 describes its structure. An *Association Business Information Entity* is derived from an *Association Core*
4814 *Component*.
- 4815 **Association Business Information Entity Property** - A *Business Information Entity Property* for which the
4816 permissible values are expressed as a complex structure, represented by an *Aggregate Business*
4817 *Information Entity*.
- 4818 **Association Core Component (ASCC)** - A *Core Component* which constitutes a complex business
4819 characteristic of a specific *Aggregate Core Component* that represents an *Object Class*. It has a unique
4820 *Business Semantic* definition. An *Association Core Component* represents an *Association Core*
4821 *Component Property* and is associated to an *Aggregate Core Component*, which describes its structure.
- 4822 **Association Core Component Property** – A *Core Component Property* for which the permissible values
4823 are expressed as a complex structure, represented by an *Aggregate Core Component*.
- 4824 **Association Type** – The association type of the *Association Business Information Entity*.
- 4825 **Attribute** – A named value or relationship that exists for some or all instances of some entity and is
4826 directly associated with that instance.
- 4827 **Basic Business Information Entity (BBIE)** – A *Business Information Entity* that represents a singular
4828 business characteristic of a specific *Object Class* in a specific *Business Context*. It has a unique *Business*
4829 *Semantic* definition. A *Basic Business Information Entity* represents a *Basic Business Information Entity*
4830 *Property* and is therefore linked to a *Data Type*, which describes its values. A *Basic Business Information*
4831 *Entity* is derived from a *Basic Core Component*.
- 4832 **Basic Business Information Entity Property** – A *Business Information Entity Property* for which the
4833 permissible values are expressed by simple values, represented by a *Data Type*.
- 4834 **Basic Core Component (BCC)** – A *Core Component* which constitutes a singular business characteristic of
4835 a specific *Aggregate Core Component* that represents an *Object Class*. It has a unique *Business Semantic*
4836 definition. A *Basic Core Component* represents a *Basic Core Component Property* and is therefore of a
4837 *Data Type*, which defines its set of values. *Basic Core Components* function as the properties of
4838 *Aggregate Core Components*.
- 4839 **Basic Core Component (CC) Property** – A *Core Component Property* for which the permissible values are
4840 expressed by simple values, represented by a *Data Type*.
- 4841 **Business Context** – The formal description of a specific business circumstance as identified by the values of
4842 a set of *Context Categories*, allowing different business circumstances to be uniquely distinguished.
- 4843 **Business Information Entity (BIE)** – A piece of business data or a group of pieces of business data with a unique
4844 *Business Semantic* definition. A *Business Information Entity* can be a *Basic Business Information Entity* (BBIE),
4845 an *Association Business Information Entity* (ASBIE), or an *Aggregate Business Information*
4846 *Entity* (ABIE).
- 4847 **Business Information Entity (BIE) Property** – A business characteristic belonging to the *Object Class* in its
4848 specific *Business Context* that is represented by an *Aggregate Business Information Entity*.

- 4849 ***Business Libraries*** – A collection of approved process models specific to a line of business (e.g., shipping, insurance).
- 4850 ***Business Process*** – The *Business Process* as described using the UN/CEFACT Modelling Methodology.
- 4851 ***Business Process Context*** – The *Business Process* name(s) as described using an appropriate list of relevant business processes.
- 4852 ***Business Process Role Context*** – The actor(s) conducting a particular *Business Process*.
- 4853 ***Business Semantic(s)*** – A precise meaning of words from a business perspective.
- 4854 ***Business Term*** – This is a synonym under which the *Core Component* or *Business Information Entity* is commonly known and used in the business. A *Core Component* or *Business Information Entity* may have several *Business Terms* or synonyms.
- 4855 ***Cardinality*** – An indication whether a characteristic is optional, mandatory and/or repetitive.
- 4856 ***CCL*** – see *Core Component Library*.
- 4857 ***Classification Scheme*** – This is an officially supported scheme to describe a given *Context Category*.
- 4858 ***Composition*** – A form of aggregation which requires that a part instance be included in at most one composite at a time, and that the composite object is responsible for the creation and destruction of the parts. *Composition* may be recursive.
- 4859 ***Content Component*** – Defines the *Primitive Type* used to express the content of a *Core Component Type*.
- 4860 ***Content Component Restrictions*** – The formal definition of a format restriction that applies to the possible values of a *Content Component*.
- 4861 ***Context*** – Defines the circumstances in which a *Business Process* may be used. This is specified by a set of *Context Categories* known as *Business Context*.
- 4862 ***Context Category*** – A group of one or more related values used to express a characteristic of a business circumstance.
- 4863 ***Controlled Vocabulary*** – A supplemental vocabulary used to define potentially ambiguous words or *Business Terms*. This ensures that every word within any of the core component names and definitions is used consistently, unambiguously and accurately.
- 4864 ***Core Component (CC)*** – A building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept.
- 4865 ***Core Component Library*** – The *Core Component Library* will contain all the *Core Component Types*, *Basic Core Components*, *Aggregate Core Components*, *Basic Business Information Entities*, *Aggregate Business Information Entities*, and *Data Types*.
- 4866 ***Core Component Property*** – A business characteristic belonging to the *Object Class* represented by an *Aggregate Core Component*.
- 4867 ***Core Component Type (CCT)*** – A *Core Component*, which consists of one and only one *Content Component*, that carries the actual content plus one or more *Supplementary Components* giving an essential extra definition to the *Content Component*. *Core Component Types* do not have *Business Semantics*.
- 4868 ***Data Type*** – Defines the set of valid values that can be used for a particular *Basic Core Component Property* or *Basic Business Information Entity Property*. It is defined by specifying restrictions on the *Core Component Type* that forms the basis of the *Data Type*.
- 4869 ***Definition*** – This is the unique semantic meaning of a *Core Component*, *Business Information Entity*, *Business Context* or *Data Type*.
- 4870 ***Dictionary Entry Name*** – This is the unique official name of a *Core Component*, *Business Information Entity*, *Business Context* or *Data Type* in the library.

| | |
|------|--|
| 4894 | <i>Geopolitical Context</i> – A combination of political and geographic factors influencing or delineating a country or region. |
| 4895 | |
| 4896 | <i>Industry Classification Context</i> – Semantic influences related to the industry or industries of the trading partners (e.g., product identification schemes used in different industries). |
| 4897 | |
| 4898 | <i>Information Entity</i> – A reusable semantic building block for the exchange of business-related information. |
| 4899 | <i>Lower-Camel-Case (LCC)</i> – a style that capitalizes the first character of each word except the first word and compounds the name. |
| 4900 | |
| 4901 | <i>Naming Convention</i> – The set of rules that together comprise how the <i>Dictionary Entry Name</i> for <i>Core Components</i> and <i>Business Information Entities</i> are constructed. |
| 4902 | |
| 4903 | <i>Object Class</i> – The logical data grouping (in a logical data model) to which a data element belongs (ISO11179). The <i>Object Class</i> is the part of a <i>Core Component's Dictionary Entry Name</i> that represents an activity or object in a specific <i>Context</i> . |
| 4904 | |
| 4905 | |
| 4906 | <i>Object Class Term</i> – A component of the name of a <i>Core Component</i> or <i>Business Information Entity</i> which represents the <i>Object Class</i> to which it belongs. |
| 4907 | |
| 4908 | <i>Official Constraints Context</i> – Legal and governmental influences on semantics (e.g. hazardous materials information required by law when shipping goods). |
| 4909 | |
| 4910 | <i>Primitive Type</i> – Used for the representation of a value. Possible values are String, Decimal, Integer, Boolean, Date and Binary. |
| 4911 | |
| 4912 | <i>Product Classification Context</i> – Factors influencing semantics that are the result of the goods or services being exchanged, handled, or paid for, etc. (e.g. the buying of consulting services as opposed to materials) |
| 4913 | |
| 4914 | <i>Property</i> – A peculiarity common to all members of an <i>Object Class</i> . |
| 4915 | |
| 4916 | <i>Property Term</i> – A semantically meaningful name for the characteristic of the <i>Object Class</i> that is represented by the <i>Core Component Property</i> . It shall serve as basis for the <i>Dictionary Entry Name</i> of the <i>Basic and Association Core Components</i> that represents this <i>Core Component Property</i> . |
| 4917 | |
| 4918 | <i>Qualifier Term</i> – A word or group of words that help define and differentiate an item (e.g. a <i>Business Information Entity</i> or a <i>Data Type</i>) from its associated items (e.g. from a <i>Core Component</i> , a <i>Core Component Type</i> , another <i>Business Information Entity</i> or another <i>Data Type</i>). |
| 4919 | |
| 4920 | |
| 4921 | <i>Registry Class</i> – The formal definition of all the information necessary to be recorded in the Registry about a <i>Core Component</i> , a <i>Business Information Entity</i> , a <i>Data Type</i> or a <i>Business Context</i> . |
| 4922 | |
| 4923 | <i>Representation Term</i> – The type of valid values for a <i>Basic Core Component</i> or <i>Business Information Entity</i> . |
| 4924 | |
| 4925 | <i>Supplementary Component</i> – Gives additional meaning to the <i>Content Component</i> in the <i>Core Component Type</i> . |
| 4926 | |
| 4927 | <i>Supplementary Component Restrictions</i> – The formal definition of a format restriction that applies to the possible values of a <i>Supplementary Component</i> . |
| 4928 | |
| 4929 | <i>Supporting Role Context</i> – Semantic influences related to non-partner roles (e.g., data required by a third-party shipper in an order response going from seller to buyer.) |
| 4930 | <i>Syntax Binding</i> – The process of expressing a <i>Business Information Entity</i> in a specific syntax. |
| 4931 | |
| 4932 | <i>System Capabilities Context</i> – This <i>Context category</i> exists to capture the limitations of systems (e.g. an existing back office can only support an address in a certain form). |
| 4933 | |
| 4934 | <i>UMM Information Entity</i> – A <i>UMM Information Entity</i> realizes structured business information that is exchanged by partner roles performing activities in a business transaction. Information entities include or reference other information entities through associations.” |
| 4935 | |
| 4936 | <i>Unique Identifier</i> – The identifier that references a <i>Registry Class</i> instance in a universally unique and unambiguous way. |
| 4937 | |
| 4938 | <i>Upper-Camel-Case (UCC)</i> – a style that capitalizes the first character of each word and compounds the name. |
| 4939 | |

- 4940 *Usage Rules* – *Usage Rules* describe how and/or when to use the *Registry Class*.
- 4941 *User Community* – A *User Community* is a group of practitioners, with a publicised contact address, who
4942 may define *Context* profiles relevant to their area of business. Users within the community do not create,
4943 define or manage their individual *Context* needs but conform to the community's standard. Such a
4944 community should liaise closely with other communities and with general standards-making bodies to avoid
4945 overlapping work. A community may be as small as two consenting organisations.
- 4946 *Version* – An indication of the evolution over time of an instance of a *Core Component*, *Data Type*, *Business*
4947 *Context*, or *Business Information Entity*.
- 4948 *XML schema* – A Recommendation of the World Wide Web Consortium (W3C), which specifies how to
4949 formally describe the elements in an Extensible Markup Language (XML) document. This description can
4950 be used to verify that each item of content in a document adheres to the description of the element in
4951 which the content is to be placed.
- 4952

4953 Intellectual Property Disclaimer

4954 ECE draws attention to the possibility that the practice or implementation of its outputs (which include but are
4955 not limited to Recommendations, norms, standards, guidelines and technical specifications) may involve the
4956 use of a claimed intellectual property right.

4957 Each output is based on the contributions of participants in the UN/CEFACT process, who have agreed to
4958 waive enforcement of their intellectual property rights pursuant to the UN/CEFACT IPR Policy (document
4959 ECE/TRADE/C/CEFACT/2010/20/Rev.2 available at http://www.unece.org/cefact/cf_docs.html or from the
4960 ECE secretariat). ECE takes no position concerning the evidence, validity or applicability of any claimed
4961 intellectual property right or any other right that might be claimed by any third parties related to the
4962 implementation of its outputs. ECE makes no representation that it has made any investigation or effort to
4963 evaluate any such rights.

4964 Implementers of UN/CEFACT outputs are cautioned that any third-party intellectual property rights claims
4965 related to their use of a UN/CEFACT output will be their responsibility and are urged to ensure that their use
4966 of UN/CEFACT outputs does not infringe on an intellectual property right of a third party.

4967 ECE does not accept any liability for any possible infringement of a claimed intellectual property right or any
4968 other right that might be claimed to relate to the implementation of any of its outputs.