



XML Naming and Design Rules

Draft 1.1a, 16 February 2005

1 Status of this Documents

This UN/CEFACT Technical Specification has been developed in accordance with the UN/CEFACT/TRADE/22 Open Development Process (ODP) for Technical Specifications. It has been approved by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) Applied Techniques Group (ATG) for implementation verification in accordance with Step 6 of the ODP.

Distribution of this document is unlimited.

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

This version:

XML Naming and Design Rules, Version 1.1a of 16 February 2005

Previous version:

NamingAndDesignRules_1.1.doc

Document identifier:

NamingAndDesignRules_1.1a.doc

Location:

<http://www.disa.org/cefact-groups/atg/downloads/index.cfm>

2 UN/CEFACT – XML Naming and Design Rules Project Team Participants

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

Project Team Leader:

Mark Crawford LMI

Editors:

Gunther Stuhec SAP AG

Paula Heilig Worldspan

Margaret Pemberton Diskray

Garret Minakawa Oracle/OAGI

Contributors:

Hisanao Sugamata ECOM-Japan

Frank Lin GCOM

K.K. Suen EAN Hong Kong

Luc Mouchot CNAM-TS

Thomas Bikeev EAN.UCC

Jostein Frømyr EDISYS

Sue Probert SEPIA eb

Alain Dechamps CEN

Michael Dill GEFEG

2.1 Acknowledgements

The UN/CEFACT - *XML Naming and Design Rules* were developed in close coordination with other XML standards efforts. In particular, the *OASIS Universal Business Language Technical Committee Naming and Design Rules* were instrumental in developing this document. Additionally, contributions were also received from:

SWIFT

U.S. Department of the Navy

U.S. Environmental Protection Agency

U.S. Federal CIO Council XML Working Group

OpenTravel Alliance

Australia Electricity & Gas Industry

CIDX

EAN/UCC

European Transmission System Operators

PIDX

38 **3 Table of Contents**

39	1	STATUS OF THIS DOCUMENTS	2
40	2	UN/CEFACT – XML NAMING AND DESIGN RULES PROJECT TEAM PARTICIPANTS. 3	
41	2.1	Acknowledgements	3
42	3	TABLE OF CONTENTS	4
43	4	INTRODUCTION	8
44	4.1	Scope and Focus.....	8
45	4.2	Audience	8
46	4.3	Structure of this Specification	8
47	4.4	Terminology and Notation	9
48	4.5	Related Documents	9
49	4.6	Conformance.....	9
50	4.7	Guiding Principles	9
51	5	GENERAL XML CONSTRUCT	11
52	5.1	Overall Schema Structure.....	11
53	5.2	Relationship to the CCTS	11
54	5.2.1	CCTS	11
55	5.2.2	Business Information Entities.....	12
56	5.2.3	The XML Constructs	13
57	5.3	Naming and Modelling Constraints	15
58	5.4	Reusability Scheme.....	16
59	5.4.1	Element Naming Conventions.....	17
60	5.5	Modularity Model.....	17
61	5.5.1	Root Schema.....	18
62	5.5.2	Internal Schema	19
63	5.5.3	External Schema.....	19
64	5.6	Namespace Scheme.....	22
65	5.6.1	UN/CEFACT Namespace Scheme	23
66	5.6.2	Declaring Namespace	23
67	5.6.3	Namespace Persistence.....	24
68	5.6.4	Namespace Uniform Resource Identifiers	24
69	5.6.5	Namespace Constraint	25
70	5.6.6	UN/CEFACT XSD Schema Namespace Tokens	25
71	5.7	Schema Location.....	26

72	5.8	Versioning	26
73	5.8.1	Major Versions	26
74	5.8.2	Minor Versions	27
75	6	GENERAL XML SCHEMA LANGUAGE CONVENTIONS.....	29
76	6.1	Schema Construct.....	29
77	6.1.1	Constraints on Schema Construction.....	29
78	6.2	Attribute and Element Declarations	29
79	6.2.1	Attributes	29
80	6.2.2	Elements	30
81	6.3	Type Definitions.....	31
82	6.3.1	Usage of Types	31
83	6.3.2	Simple Type Definitions.....	31
84	6.3.3	Complex Type Definitions	31
85	6.4	Use of XSD Extension and Restriction.....	32
86	6.4.1	Extension	32
87	6.4.2	Restriction.....	32
88	6.5	Annotation.....	32
89	6.5.1	Documentation	32
90	7	XML SCHEMA MODULES.....	36
91	7.1	Root Schema.....	36
92	7.1.1	Schema Construct.....	36
93	7.1.2	Namespace Scheme	36
94	7.1.3	Imports and Includes	37
95	7.1.4	Root Element Declaration.....	37
96	7.1.5	Type Definitions	38
97	7.1.6	Annotations.....	38
98	7.2	Internal Schema.....	39
99	7.2.1	Schema Construct.....	39
100	7.2.2	Namespace Scheme	39
101	7.2.3	Imports and Includes	39
102	7.3	Reusable Aggregate Business Information Entities.....	39
103	7.3.1	Schema Construct.....	39
104	7.3.2	Namespace Scheme	40
105	7.3.3	Imports and Includes	40
106	7.3.4	Type Definitions	41
107	7.3.5	Element Declarations.....	43
108	7.4	Annotation.....	43
109	7.5	Core Component Type	47
110	7.5.1	Use of Core Component Type Module.....	47
111	7.5.2	Schema Construct.....	47
112	7.5.3	Namespace Scheme	47
113	7.5.4	Imports and Includes	47
114	7.5.5	Type Definitions	48
115	7.5.6	Attribute Declarations.....	48
116	7.5.7	Extension and Restriction.....	49
117	7.5.8	Annotation	49

118	7.6	Unqualified Data Type	50
119	7.6.1	Use of Unqualified Data Type Module.....	50
120	7.6.2	Schema Construct	50
121	7.6.3	Namespace Scheme	51
122	7.6.4	Imports and Includes	51
123	7.6.5	Type Definitions	52
124	7.6.6	Attribute Declarations.....	52
125	7.6.7	Restriction.....	55
126	7.6.8	Annotation	55
127	7.7	Qualified Data Type	56
128	7.7.1	Use of Qualified Data Type Module.....	56
129	7.7.2	Schema Construct	57
130	7.7.3	Namespace Scheme	57
131	7.7.4	Imports and Includes	57
132	7.7.5	Type Definitions	58
133	7.7.6	Attribute and Element Declarations.....	59
134	7.7.7	Extension and Restriction.....	59
135	7.7.8	Annotation	59
136	7.8	Code Lists	61
137	7.8.1	Schema Construct	62
138	7.8.2	Namespace Name for Code Lists	62
139	7.8.3	UN/CEFACT XSD Schema Namespace Token for Code Lists	64
140	7.8.4	Schema Location	65
141	7.8.5	Imports and Includes	65
142	7.8.6	Type Definitions	65
143	7.8.7	Element Declarations.....	66
144	7.8.8	Extension and Restriction.....	66
145	7.8.9	Annotation	67
146	7.9	Identifier List Schema	67
147	7.9.1	Schema Construct	68
148	7.9.2	Namespace Name for Identifier List Schema	68
149	7.9.3	UN/CEFACT XSD Schema Namespace Token for Identifier List Schema	70
150	7.9.4	Schema Location	70
151	7.9.5	Imports and Includes	71
152	7.9.6	Type Definitions	71
153	7.9.7	Attribute and Element Declarations.....	72
154	7.9.8	Extension and Restriction.....	72
155	7.9.9	Annotation	73
156	8	XML INSTANCE DOCUMENTS	74
157	8.1	Character Encoding	74
158	8.2	Empty Content	74
159	8.3	xsi:type	74
160	APPENDIX A. RELATED DOCUMENTS		75
161	APPENDIX B. OVERALL STRUCTURE		76
162	APPENDIX C. ATG APPROVED ACRONYMS AND ABBREVIATIONS		83
163	APPENDIX D. CORE COMPONENT SCHEMA MODULE		84

164	APPENDIX E. UNQUALIFIED DATA TYPE SCHEMA MODULE	85
165	APPENDIX F. ANNOTATION TEMPLATES	86
166	APPENDIX G. MAPPING OF CCTS REPRESENTATION TERMS TO CCT AND UDT DATA TYPES..	90
167	APPENDIX H. NAMING & DESIGN RULES LIST	91
168	APPENDIX I. GLOSSARY	106
169	APPENDIX J. QUALIFIED DATA TYPE SCHEMA MODULE.....	110

170 **4 Introduction**

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

173 This technical specification provides a way to identify, capture and maximize the re-use of business
174 information expressed as XML schema components to support and enhance information interoperability
175 across multiple business situations.

176 **4.1 Scope and Focus**

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

181 This technical specification will form the basis for standards development work of technical experts
182 developing XML schema based on information models developed in accordance with the UN/CEFACT
183 *Core Components Technical Specification – Part 8 of the ebXML Framework (CCTS)*. The CCTS
184 specification has subsequently been published as ISO/TS 15000-5 ebCCTS ebXML *Electronic Business*
185 *Extensible Mark-up Language, Part 5: ebCCTS ebXML Core Components Technical Specification,*
186 *Version 2.01 (2003-11-15)*.

187 **4.2 Audience**

188 The primary audience for this UN/CEFACT – *XML Naming and Design Rules* Technical Specification are
189 members of the UN/CEFACT Applied Technologies Group who are responsible for development and
190 maintenance of UN/CEFACT XML schema. The intended audience also includes the wider membership
191 of the other UN/CEFACT Groups who will participate in the process of creating and maintaining
192 UN/CEFACT XML schema.

193 Additional audiences are designers of tools who need to specify the conversion of user input into XML
194 schema representation adhering to the rules defined in this document. Additionally, designers of XML
195 schema outside of the UN/CEFACT Forum community may find the rules contained herein suitable as
196 design rules for their own organization.

197 **4.3 Structure of this Specification**

198 The UN/CEFACT *XML Naming and Design Rules* Technical Specification has been divided into 5 main
199 sections.

200 Section 4 provides general information about the document itself as well as normative statements in
201 respect to conformance.

202 Section 5 provides information on the guiding principles applied in developing this specification as well as
203 its dependency and relationship to CCTS. Furthermore, this section describes the approach taken to
204 modularity in order to maximize the re-use of business information expressed as XML schema
205 components and the general naming conventions applied. (Normative)

206 Section 6 provides the general conventions applied with respect to the use of the XML schema language.
207 (Normative)

208 Section 7 provides detailed rules applicable to each of the schema modules defined by the modularity
209 approach. (Normative)

210 Section 8 provides guidelines and rules related to XML instance documents. (Normative)

211 The document also contains the following Appendices:

212 Appendix A Related Documents (Informative)

213 Appendix B Overall Structure (Normative)

214 Appendix C ATG Approved Acronyms and Abbreviations (Normative)

- 215 Appendix D Core Component Schema Module (Normative)
- 216 Appendix E Unqualified Data Type Schema Module (Normative)
- 217 Appendix F Annotation Templates (Informative)
- 218 Appendix G Mapping of CCTS Representation Terms to CCT and UDT Data Types (Informative)
- 219 Appendix H Naming and Design Rules List (Normative)
- 220 Appendix I Glossary (Informative)
- 221 Appendix J Qualified Data Type Schema Module (Normative)

222 4.4 Terminology and Notation

223 The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT,
224 RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as
225 described in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2119.¹ Wherever
226 xsd: appears this refers to a construct taken from the W3C XML schema specification. Wherever ccts:
227 appears this refers to a construct taken from the CCTS.

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

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

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

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

234 When defining rules the following annotations are used:

- 235 • [] = optional
- 236 • < > = Variable
- 237 • | = choice

238 4.5 Related Documents

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

240 4.6 Conformance

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

243 [R 1] Conformance shall be determined through adherence to the content of normative sections,
244 rules and definitions.

245 4.7 Guiding Principles

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

- 247 • Relationship to UMM – UN/CEFACT XSD Schema will be based on UMM metamodel adherent
248 Business Process Models.
- 249 • Relationship to Information Models – UN/CEFACT XSD Schema will be based on information
250 models developed in accordance with the UN/CEFACT – Core Components Technical
251 Specification.
- 252 • Schema Creation– UN/CEFACT XML design rules will support schema creation through
253 handcrafting as well as automatic generation.

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>

- 254 • ebXML Use – UN/CEFACT XSD Schema and instance documents shall be straightforwardly
255 usable within the ebXML framework and compatible with other frameworks to the maximum
256 extent practicable.
- 257 • Interchange and Application Use – UN/CEFACT XSD Schema and instance documents are
258 intended for business-to-business and application-to-application use.
- 259 • Tool Use and Support - The design of UN/CEFACT XSD Schema will not make any assumptions
260 about sophisticated tools for creation, management, storage, or presentation being available.
- 261 • Legibility - UN/CEFACT XML instance documents should be intuitive and reasonably clear in the
262 context for which they are designed.
- 263 • Schema Features - The design of UN/CEFACT XSD Schema should use the most commonly
264 supported features of W3C XSD Schema.
- 265 • Technical Specifications – UN/CEFACT XML design rules will be based on Technical
266 Specifications holding the equivalent of W3C recommended status.
- 267 • Schema Specification – UN/CEFACT XML design rules will be fully conformant with W3C XML
268 Schema Definition Language.
- 269 • Interoperability - The number of ways to express the same information in a UN/CEFACT XSD
270 Schema and UN/CEFACT XML instance document is to be kept as close to one as possible.
- 271 • Maintenance – The design of UN/CEFACT XSD Schema must facilitate maintenance.
- 272 • Context Sensitivity - The design of UN/CEFACT XSD Schema must ensure that context-sensitive
273 document types aren't precluded.
- 274 • Relationship to Other Namespaces - UN/CEFACT XML design rules will be cautious about
275 making dependencies on other namespaces.
- 276 • Legacy formats - UN/CEFACT XML design rules are not responsible for sustaining legacy
277 formats.
- 278 • Messages must express semantics fully in schema and not rely on well-formedness.

279 5 General XML Construct

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

- 281 • Overall Schema Structure
- 282 • Relationship to CCTS
- 283 • Naming and Modelling Constraints
- 284 • Reusability Scheme
- 285 • Modularity Strategy
- 286 • Namespace Scheme
- 287 • Versioning Scheme

288 5.1 Overall Schema Structure

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

293 [R 2] All UN/CEFACT XSD Schema design rules MUST be based on the *W3C XML Schema*
294 *Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Data Types*.

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

298 [R 3] All UN/CEFACT XSD Schema and UN/CEFACT conformant XML instance documents MUST
299 be based on the W3C suite of technical specifications holding recommendation status.

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

302 [R 4] UN/CEFACT XSD Schema MUST follow the standard structure defined in Appendix B.

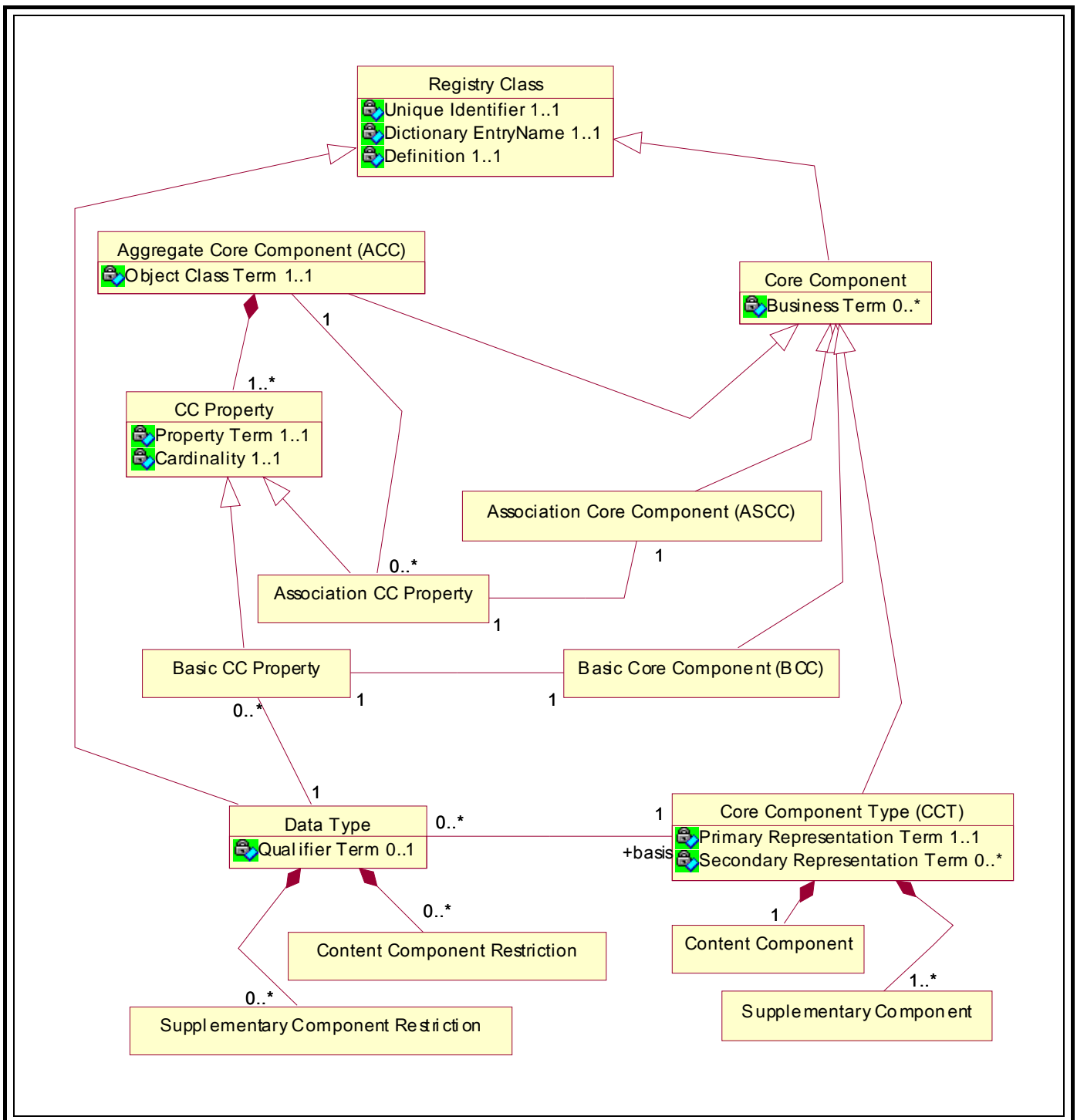
303 5.2 Relationship to the CCTS

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

306 5.2.1 CCTS

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

² *Core Components Technical Specification, Part 8 of the ebXML Technical Framework Version 2.0 (Second Edition)*, UN/CEFACT, 15 November 2003



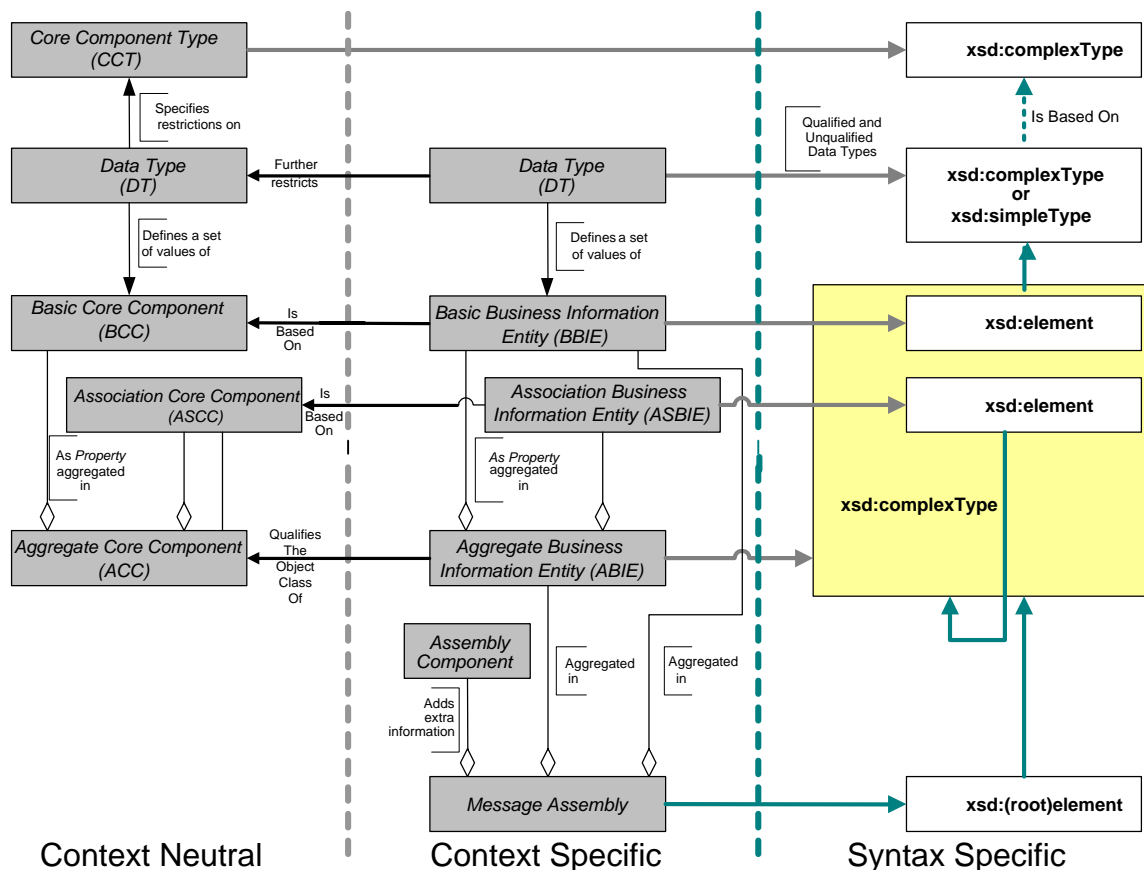
313

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

315 **5.2.2 Business Information Entities**

316 In the CCTS model, context neutral core components are instantiated as context specific components for
 317 message assembly and model harmonization. The context specific components are defined as Business
 318 Information Entities (`ccts:BusinessInformationEntities`). See CCTS Section 6.2 for a detailed
 319 discussion of the ebXML context mechanism.³ Context specific
 320 `ccts:BusinessInformationEntities` are defined in CCTS as “A piece of business data or a group

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



333

334

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

335

336

337

338

- The message assembly is represented as a `xsd:complexType` definition and global element declaration in an UN/CEFACT XSD Schema. The global element declaration is based on (is of type) `xsd:complexType` that represents the document level ABIE. The global element appears in, and is designated as the root element of, UN/CEFACT conformant XML instances.

339

- An ABIE is defined as a `xsd:complexType`.

340

341

342

343

344

- An ASBIE is declared as a local element within the `xsd:complexType` representing the associating ABIE. The ASBIE element is in itself based on (is of type) `xsd:complexType` of the associated ABIE. In this way the content model of the associated ABIE is included in the content model of the associating ABIE.

345

[Note]

346

347

348

349

350

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

351

352

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

353

354

355

356

357

358

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

359
360
361
362
363
364

[Note]

Data Types are not derived from the CCT complex types using `xsd:restriction` because whereas all CCTs are defined as complex types with attributes representing their supplementary components, in several cases we leverage built-in `xsd:simpleType` whose facets correspond to the supplementary components. See Section 7.5 for more information.

365
366
367
368

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

369

5.3 Naming and Modelling Constraints

370
371
372
373
374
375
376

UN/CEFACT XSD Schema are derived from CCTS and UN/CEFACT Modelling Methodology (UMM) process modelling and data analysis. The UN/CEFACT library contains fully conformant CCTS dictionary entry names as well as truncated XML element names developed in conformance with the naming constraint rules specified below. The XML fully qualified XPath ties the information to its standardized semantics as described in the underlying CCTS construct and CCTS Dictionary Entry Name, while the XML element or attribute name is a truncation that reflects the hierarchy inherent in the XML construct. There are differences in the rules for naming of elements, attributes, and types.

377
378

[R 5] Each element or attribute XML name MUST have one and only one fully qualified XPath (FQXP).

379
380

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

381
382
383

Example 5-1: Fully Qualified XPath

```
Address/Coordinate/Latitude Measure  
Organisation/Location/Name
```

384
385
386
387

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

388
389

[R 6] 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.

390
391
392
393
394

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

395

[R 7] Lower camel case (LCC) MUST be used for naming attributes.

396
397

Example 5-2: Attribute

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

398

[R 8] Upper camel case (UCC) MUST be used for naming elements and types.

399
400

Example 5-3: Element

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

401
402

Example 5-4: Type

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

403
404

[R 9] Element, attribute and type names MUST be in singular form unless the concept itself is plural.

405

Example 5-5: Singular and Plural Concept Form

406 **Allowed - Singular:**

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

408 **Not Allowed - Plural:**

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

410 [R 10] Element, attribute and type names MUST be drawn from the following character set: **a-z**
411 and **A-Z**.

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

413 **Not Allowed**

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

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

418 [R 11] XML element, attribute and type names constructed from dictionary entry names MUST NOT
419 include periods, spaces, or other separators; or characters not allowed by W3C XML 1.0 for
420 XML names.

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

422 **Not Allowed**

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

424 [R 12] XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other
425 word truncations, except those included in the UN/CEFACT controlled vocabulary or listed in
426 Appendix C.

427 [R 13] Acronyms and abbreviations at the beginning of an attribute declaration MUST appear in all
428 lower case. All other acronym and abbreviation usage in an attribute declaration must
429 appear in upper case.

430 [R 14] Acronyms MUST appear in all upper case for all element declarations and type definitions.

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

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

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

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

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

437 **5.4 Reusability Scheme**

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

453
454

[R 15] All element declarations for BBIEs and ASBIEs MUST be locally declared within the parent ABIE type.

455

5.4.1 Element Naming Conventions

456
457
458
459
460
461

The fully qualified XPath anchors the use of a construct to a particular location in a business message. The dictionary definition identifies any semantic dependencies that the FQXP has on other elements and attributes within the UN/CEFACT library that are not otherwise enforced or made explicit in its structural definition. The dictionary serves as a traditional data dictionary, and also serves some of the functions of traditional implementation guides. As discussed in Section 5.4 above, the dictionary must be carefully controlled to overcome the limitations in control inherent in a local element approach.

462

5.5 Modularity Model

463
464
465
466

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

467
468
469
470
471
472
473

Accordingly UN/CEFACT has defined a number of schema modules to support this approach. Figure 5-4 portrays the CEFACT modularity model. We categorize our modules into message assembly and external schema. The message assembly consists of root schema and internal schema modules that reside in the same namespace as the root schema. The external schema modules consist of a set of reusable schema for ABIEs, unqualified data types, qualified data types, code lists and identifier lists. Each of these schema modules reside in their own namespace. Dependencies exist amongst the various modules as shown in the figure.

474
475
476
477
478
479

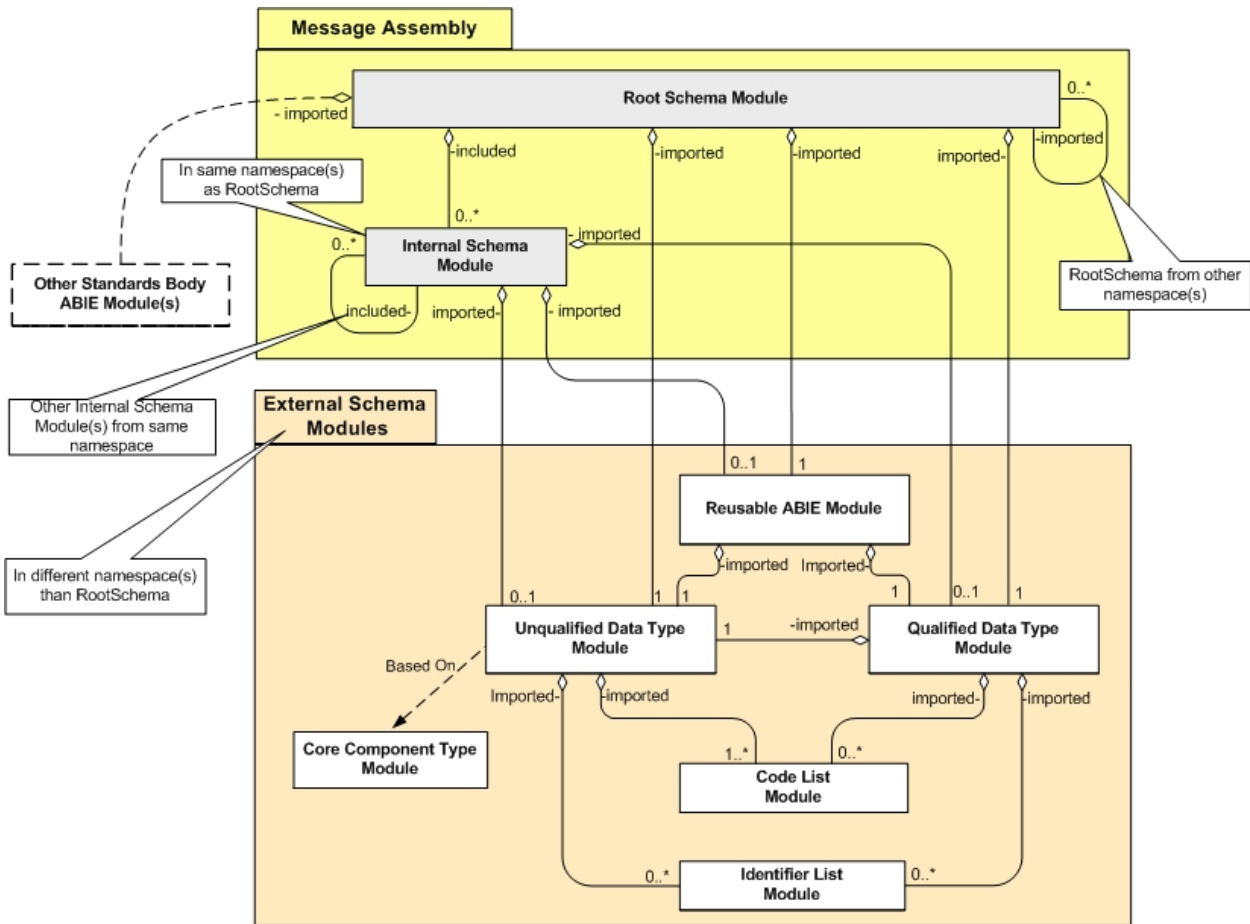
The root schema always includes any internal schema residing in its namespace. It also always imports the ABIE reusable schema, unqualified and qualified data type schema modules. It may import root schemas from other namespaces as well as reusable schema from other standards bodies. The internal schema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

480
481
482
483

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

484
485

The core component type schema module is provided as reference documentation and is used as the basis for the unqualified data type schema module.



486

487 **Figure 5-4 UN/CEFACT XSD Schema Modularity Scheme**

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

Schema Module Name	Token
RootSchema	rsm
CCTS CCT	cct
UN/CEFACT Reusable Aggregate Business Information Entity	ram
UN/CEFACT Unqualified Data Type	udt
UN/CEFACT Qualified Data Type	qdt
Code List	clm
Identifier List	ids

491 **5.5.1 Root Schema**

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

495 [R 16] A root schema MUST be created for each unique business information exchange.

496 The UN/CEFACT modularity approach enables the reuse of individual root schema without having to
 497 import the entire UN/CEFACT root schema library. Additionally, a root schema can import individual
 498 modules without having to import all UN/CEFACT XSD schema modules. Each root schema will define
 499 its own dependencies. A root schema should not duplicate reusable XML constructs contained in other
 500 schema, rather it should reuse existing constructs available elsewhere. Specifically, root schema will

501 import or include other schema modules to maximize reuse through `xsd:include` or `xsd:import` as
502 appropriate.

503 [R 17] A root schema MUST NOT replicate reusable constructs available in schema modules
504 capable of being referenced through `xsd:include` or `xsd:import`.

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

507 [R 18] UN/CEFACT XSD schema modules MUST either be treated as external schema modules or
508 as internal schema modules of the root schema.

509 5.5.2 Internal Schema

510 Not all ABIEs will be suitable for widespread reuse. Some may be limited to a specific business function
511 or information exchange. These ABIEs will be defined as `xsd:complexType` in an internal schema
512 module rather than in the reusable ABIE module, (See Section 5.5.3.4 below). UN/CEFACT XSD
513 Schema may have zero or more internal modules.

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

519 [R 19] All UN/CEFACT internal schema modules MUST be in the same namespace as their
520 corresponding `rsm:RootSchema`.

521 UN/CEFACT internal schema modules will necessarily have a semantically meaningful name. Internal
522 schema module names will identify the parent root schema module, the internal schema module function,
523 and the schema module itself.

524 [R 20] Each UN/CEFACT internal schema module MUST be named
525 <ParentRootSchemaModuleName><InternalSchemaModuleFunction> Schema Module

526 Example 5-9: UN/CEFACT internal schema module name

527 `TravelReservationRequestFlightInformation Schema Module`
528 Where:
529 `TravelReservationRequest` represents the parent root schema module name
530 `FlightInformation` represents the internal schema module function

531 5.5.3 External Schema

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

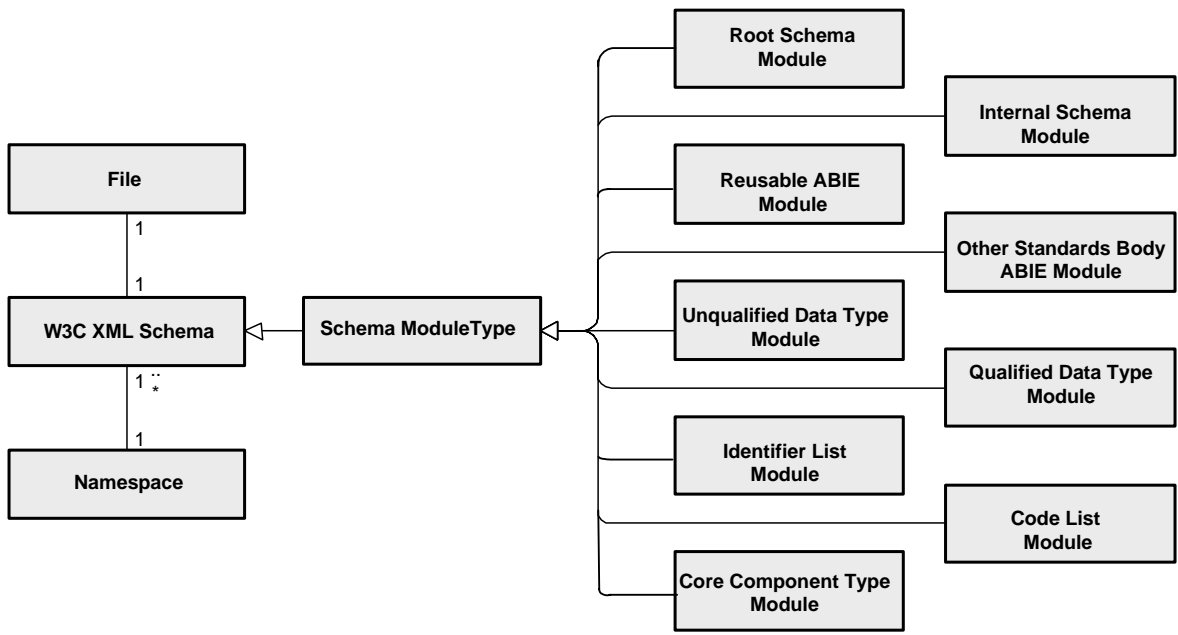
- 536 • Core Component Type
 - 537 • Unqualified Data Type
 - 538 • Qualified Data Type
 - 539 • Reusable ABIE
 - 540 • Code List
 - 541 • Identifier List
 - 542 • Other Standards Body ABIE module
- 543

544 [Note]

545
546
547

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

548 These external schema modules are reflected in Figure 5-5.



549
550

Figure 5-5 UN/CEFACT XSD Schema Modules

5.5.3.1 Core Component Type Schema Module

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

555 [R 21] A Core Component Type schema module MUST be created

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

558 [R 22] The `cct:CoreComponentType` schema module MUST be named "CCTS CCT Schema
559 Module"

560 The current version of the normative UN/CEFACT CCT schema module is contained in Appendix D.

5.5.3.2 Unqualified Data Type Schema Module

562 A schema module is required to represent the normative form data types for each CCT as expressed in
563 the CCTS meta model. These data types are based on the XSD constructs from the CCT schema
564 module but where possible reflect the use of built-in `xsd:simpleType` rather than their parent CCT
565 `xsd:complexType`. As such, the unqualified data type schema module does not import the CCT
566 schema module. The unqualified data types are so named because they contain no additional
567 restrictions on their source CCTs other than those defined in CCTS and agreed upon best practices. An
568 unqualified data type is defined for all approved CCTS primary and secondary representation terms.

569 [R 23] An Unqualified Data Type schema module MUST be created

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

572 [R 24] The `udt:UnqualifiedDataType` schema module MUST be named "UN/CEFACT
573 Unqualified Data Type Schema Module"

574 The current version of the normative UN/CEFACT Unqualified Data Type Schema Module is contained in
575 Appendix E.

576 **5.5.3.3 Qualified Data Type Schema Module**

577 As data types are reused for different BIEs, restrictions on the data type may be applied. These restricted
578 data types are referred to as qualified data types. These qualified data types will be defined in a separate
579 qualified data type schema module. A qualified data type schema module will import the UN/CEFACT
580 Unqualified Data Type Schema Module. In the future this single qualified data type schema module may
581 be segmented into additional modules if deemed necessary.

582 [R 25] A Qualified Data Type schema module MUST be created..

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

585 [R 26] The `qdt:QualifiedDataType` schema module MUST be named "UN/CEFACT Qualified
586 Data Type Schema Module"

587 The current version of the normative UN/CEFACT Qualified Data Type Schema Module is contained in
588 Appendix E.

589 **5.5.3.4 Reusable Aggregate Business Information Entity Schema Module**

590 A single reusable aggregate business information entity schema module is required. This schema
591 module will contain a type definition for every reusable ABIE in the UN/CEFACT Core Component Library.
592 In the future this single reusable schema module may be segmented into additional modules if deemed
593 necessary. This single reusable schema module may be compressed for run time performance
594 considerations if necessary. Compression means that a run time version of the reusable ABIE schema
595 module would be created that would consist of a subset of the ABIE constructs. This subset would
596 consist only of those ABIEs necessary to support the specific root schema being validated.

597 [R 27] A Reusable Aggregate Business Information Entity schema module MUST be created

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

600 [R 28] The `ram:ReusableAggregateBusinessInformationEntity` schema module MUST
601 be named "UN/CEFACT Reusable Aggregate Business Information Entity Schema Module"

602 **5.5.3.5 Code List Schema Modules**

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

606 [R 29] Reusable Code List schema modules MUST be created to convey code list enumerations

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

609 [R 30] The name of each `clm:CodeList` schema module MUST be of the form: `<Code List`
610 `Agency Identifier|Code List Agency Name><Code List Identification`
611 `Identifier|Code List Name> - Code List Schema Module`

612 Where:

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

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

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

616 Code List Name = The name of the code list as assigned by the agency that maintains the
617 code list
618

619 **Example 5-10: Name of UN/CEFACT Account Type Code Schema Module**

620 [64437 - Code List Schema Module](#)

```
621 where:
622 6 = Code list agency identifier for UN/CEFACT as defined in UN/CEFACT code
623 list 3055
624 4437 = Code list identification identifier for Account Type Code in UN/CEFACT
625 directory
```

626 **Example 5-11: Name for a code using agency name and code list name**
627 `Security Initiative Document Security Code - Code List Schema Module`

628 5.5.3.6 Identifier List Schema Module

629 In those cases where run time validation is required against a used identifier scheme, a separate identifier
630 list schema module will be created to minimize the impact of identifier list changes on root and other
631 reusable schema. Each reusable identifier list schema module will contain enumeration values for the
632 identifiers.

633 [R 31] An Identifier List schema module MUST be created to convey enumeration values for each
634 identifier list that requires run time validation .

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

637 [R 32] The name of each `ids:IdentifierList` schema module MUST be of the form:
638 `<Identifier Scheme Agency Identifier|Identifier Scheme Agency`
639 `Name><Identifier Scheme Identifier|Identifier Scheme Name> -`
640 `Identifier List Schema Module`

641 Where:
642 Identifier Scheme Agency Identifier = The identification of the agency that maintains the
643 identification scheme
644 Identifier Scheme Agency Name = Agency that maintains the identifier list
645 Identifier Scheme Identifier = The identification of the identification scheme
646 Identifier Scheme Name = Name as assigned by the agency that maintains the identifier
647 list
648

649 Example 5-12: Name of ISO Country Identifier schema module

```
650 53166-1 - Identifier List Schema Module
651 where:
652 5 = Code list agency identifier for ISO as defined in UN/CEFACT code
653 list 3055
654 3166-1 = Identifier scheme identifier for Two Alpha Country Identifier in
655 ISO
```

656 5.5.3.7 Other Standards Body Aggregate Business Information Entity Schema 657 Modules

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

662 [R 33] Imported schema modules MUST be fully conformant with the *UN/CEFACT XML Naming and*
663 *Design Rules Technical Specification* and the *Core Components Technical Specification*.

664 5.6 Namespace Scheme

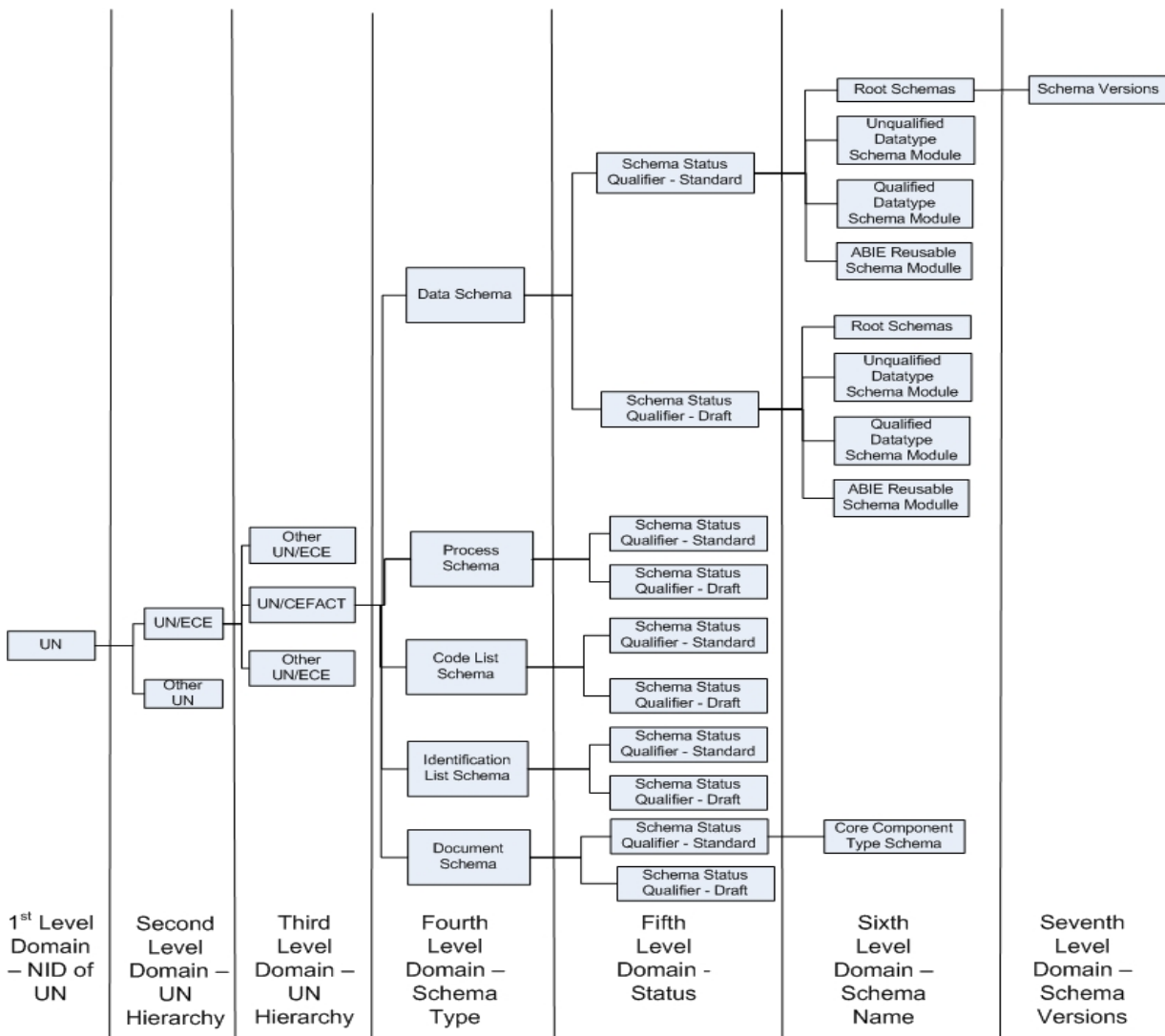
665 As defined in the W3C specification, "XML namespaces provide a simple method for qualifying element
666 and attribute names used in Extensible Markup Language documents by associating them with
667 namespaces identified by URI references."⁵ This enables interoperability and consistency in the XML
668 artefacts for the extensive library of reusable types and schema modules. The UN/CEFACT reusability

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

669 methodology maximizes the reuse of defined named types and locally declared elements and attributes
 670 within those types (See Section 5.4). In addition, the modularity approach of multiple reusable schema
 671 modules (See Section 5.5) prescribe just such a method. There exists specific relationships between the
 672 various internal and external schema modules identified in Section 5.5 with respect to their namespaces.
 673 These relationships are defined in Figure 5-4. Accordingly, a sufficiently robust namespace scheme is
 674 essential.

675 5.6.1 UN/CEFACT Namespace Scheme

676 In establishing a UN/CEFACT approach to namespaces, it is important to recognize that in addition to
 677 XML requirements, many other requirements exist for a standardized namespace approach. Accordingly,
 678 a master UN/CEFACT namespace scheme must be sufficiently flexible and robust to accommodate both
 679 XML and other syntax requirements. Figure 5-6 reflects such an approach and will be used as the basis
 680 for determining the namespace structure and rules that follow.



681
 682 **Figure 5-6: UN/CEFACT Namespace Scheme**

683 5.6.2 Declaring Namespace

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

686 [R 34] Every UN/CEFACT defined or imported schema module MUST have a namespace declared,
 687 using the `xsd:targetNamespace` attribute.

688 5.6.3 Namespace Persistence

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

700 [R 35] Every version of a defined or imported schema module other than internal schema modules
701 MUST have its own unique namespace.

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

705 [R 36] UN/CEFACT published namespace declarations or contents MUST never be changed.

706 5.6.4 Namespace Uniform Resource Identifiers

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

713 [R 37] UN/CEFACT namespaces MUST be defined as Uniform Resource Names

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

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

719 where :

720 `<NID>` = the Namespace Identifier

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

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

723 The Namespace ID determines the syntactic interpretation of the Namespace Specific String

724 Following this pattern, the UN/CEFACT namespace general structure for a namespace name should be:

725 `urn:un:unece:uncefact:<schematype>:<status>:<name>:<version>`

726 Where:

- 727 • Namespace Identifier (NID) = un
- 728 • Namespace Specific String =
- 729 • `unece:uncefact:<schematype>:<status>:<name>:<version>` with `unece` and `uncefact` as fixed
730 value second and third level domains within the NID of un
- 731 • `schematype` = a token identifying the type of schema module:
732 `data | process |odelist | identifierlist | documentation`
- 733 • `status` = the status of the schema as: `draft | standard`

- 734 • name = the name of the module (using upper camel case)
- 735 • version = <major>.<minor>.[<revision>]
- 736 • major = The major version number. Sequentially assigned, first release starting with the number
- 737 1.
- 738 • minor = The minor version number within a major release. Sequentially assigned, first release
- 739 starting with the number 0. Not applicable for code list or identifier list schema.
- 740 • revision = Sequentially assigned alphanumeric character for each revision of a minor release.
- 741 Only applicable where status = draft. Not applicable for code list or identifier list schema.

742 [R 38] The names for namespaces MUST have the following structure while the schema is at draft

743 status:

744 `urn:un:unece:uncefact:<schematype>:draft:<name>:<major>.[<minor>].[<`

745 `revision]`

746

747 Where:

748 schematype = a token identifying the type of schema module:

749 `data | process | codelist | identifierlist | documentation`

750 name = the name of the module (using upper camel case)

751 major = the major version number. Sequentially assigned, first release starting with the

752 number 1.

753 minor = the minor version number within a major release. Sequentially assigned, first release

754 starting with the number 0. Not applicable for code list or identifier list schema.

755 revision = sequentially assigned alphanumeric character for each revision of a minor release.

756 Only applicable where status = draft and schema type does not equal code list or

757 identifier list.

758 **Example 5-13: Namespace Name at Draft Status**

759 `"urn:un:unece:uncefact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:0.3`

760 `.5"`

761 [R 39] The namespace names for schema holding specification status MUST be of the form:

762 `urn:un:unece:uncefact:<schematype>:standard:<name>:<major>.[<minor>]`

763

764 Where:

765 schematype = a token identifying the type of schema module:

766 `data | process | codelist | identifierlist | documentation`

767 name = the name of the module

768 major = the major version number, sequentially assigned, first release starting with the

769 number 1.

770 minor = the minor version number within a major release, sequentially assigned, first release

771 starting with the number 0. Not applicable for code list or identifier list schema.

772 **Example 5-14: Namespace Name at Specification Status**

773 `"urn:un:unece:uncefact:data:standard:UNCEFACTUnqualifiedDataTypeSchemaModule:`

774 `1.0"`

775 **5.6.5 Namespace Constraint**

776 To ensure consistency in declaring namespaces, a namespace should only be declared for an XML

777 construct by the owner of that namespace – unless specifically designed as a generic namespace such

778 as xsi. Accordingly, UN/CEFACT namespaces will only contain XML constructs created and assigned by

779 UN/CEFACT.

780 [R 40] UN/CEFACT namespaces MUST only contain UN/CEFACT developed schema modules.

781 **5.6.6 UN/CEFACT XSD Schema Namespace Tokens**

782 A unique token will be defined for each namespace. The exact token for each type of namespace will be

783 defined by the applicable schema module subsection in Section 7.

784 5.7 Schema Location

785 Schema locations are required to be in the form of a URI scheme. Schema locations are typically the
786 same as their namespaces. Schema locations are typically defined as URL based URI schemes because
787 of resolvability limitations of URN based URI schemes. However, UN/CEFACT XSD Schema use a URN
788 based URI scheme for namespace declarations because persistence is considered more important than
789 resolvability. In recognition of the need for resolvability of schema location, until such time as URNs
790 become fully resolvable, UN/CEFACT will store schema in locations identified using a URL based URI
791 scheme aligned with the URN based URI scheme used for the namespace declaration as follows:

792 urn:un:unece:uncefact:<schematype>:<status>:<name>:<version>

793 [R 41] The general structure for schema location MUST be:
794 `http://www.unece.org/uncefact/<schematype>/<name>_<major>.<minor>.`
795 `[<revision>]_[<status>].xsd`

796 Where:

797 schematype = a token identifying the type of schema module:

798 `data | process | codelist | identifierlist | documentation`

799 name = the name of the module (using upper camel case)

800 major = the major version number, sequentially assigned, first release starting with the
801 number 1.

802 minor = the minor version number within a major release, sequentially assigned, first release
803 starting with the number 0.

804 revision = sequentially assigned alphanumeric character for each revision of a minor release.

805 Only applicable where status = draft.

806 status = the status of the schema as: `draft | standard`
807

808 [R 42] Each `xsd:schemaLocation` attribute declaration MUST contain a persistent and resolvable
809 URL.

810 [R 43] Each `xsd:schemaLocation` attribute declaration URL MUST contain an absolute path.

811 5.8 Versioning

812 A UN/CEFACT namespace URN is divided into three parts. First is the standard UN/CEFACT namespace
813 information. Second is the description of the purpose of the namespace. Third is the version information.
814 The version information will in turn be divided into major (or incompatible) and minor (or compatible)
815 fields. The minor field has an optional revision extension.

816 5.8.1 Major Versions

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

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

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

827 urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>.0

828 Where:

- 829 • major = the first release starts with the number 1.
- 830 • minor = always 0 for major release numbers.

831 [R 44] Every schema major version MUST have the URI of:
832 `urn:un:unece:unefact:<schematype>:<status>:<name>:<major>.0.[<revis`
833 `ion>]`

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

837 [R 45] Every UN/CEFACT XSD Schema and schema module major version number MUST be a
838 sequentially assigned incremental integer greater than zero.

839 5.8.2 Minor Versions

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

- 845 • Adding values to enumerations
- 846 • Optional extensions
- 847 • Add optional elements

848 [R 46] Minor versioning MUST be limited to declaring new optional XSD constructs, extending
849 existing XSD constructs and refinements of an optional nature.

850 Minor version numbers are reflected in the namespace declaration as follows:

851 `urn:un:unece:unefact:<schematype>:<status>:<name>:<major>.<non-zero integer>.[<revision>]`

852 Where:

- 853 • major = the major version number, sequentially assigned, first release starting with the number 1
- 854 • minor = always positive integer

855 [R 47] Every UN/CEFACT XSD Schema minor version MUST have the URI of:
856 `urn:un:unece:unefact:cc:schema:<name>:<major>.<non-zero`
857 `integer>.[<revision>]`

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

861 Minor version changes are not allowed to break compatibility with previous minor versions. Compatibility
862 includes consistency in naming of the schema constructs. UN/CEFACT minor version changes will not
863 include renaming the XML construct.

864 [R 48] For UN/CEFACT minor version changes, the name of the schema construct MUST NOT
865 change.

866 Semantic compatibility across minor versions is essential.

867 [R 49] Changes in minor versions MUST NOT break semantic compatibility with prior versions.

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

873 [R 50] UN/CEFACT minor version schema MUST incorporate all XML constructs from the
874 immediately preceding major or minor version schema.

875
876
877
878
879
880

[Note]

There has been much discussion surrounding the issue of namespaces and versioning. ATG solicits input from interested parties on the pro's and con's of assigning a unique namespace for each minor version as opposed to assigning a new namespace for only major versions and having all minor versions have the same namespace as its major version.

881 6 General XML Schema Language Conventions

882 6.1 Schema Construct

883 [R 51] The `xsd:elementFormDefault` attribute MUST be declared and its value set to “qualified”.

884 [R 52] The `xsd:attributeFormDefault` attribute MUST be declared and its value set to
885 “unqualified”.

886 [R 53] The “xsd” prefix MUST be used in all cases when referring to
887 <http://www.w3.org/2001/XMLSchema> as follows:
888 `xmlns:xsd=http://www.w3.org/2001/XMLSchema`

889 Example 6-1: Element and Attribute Form Default

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

893 6.1.1 Constraints on Schema Construction

894 [R 54] The xsi prefix SHALL be used where appropriate for referencing `xsd:schemaLocation` and
895 `xsd:noNamespaceLocation` attributes in instance documents.

896 [R 55] `xsd:appInfo` MUST NOT be used.

897 [R 56] `xsd:notation` MUST NOT be used.

898 [R 57] `xsd:wildcard` MUST NOT be used.

899 [R 58] The `xsd:any` element MUST NOT be used.

900 [R 59] The `xsd:any` attribute MUST NOT be used.

901 [R 60] Mixed content MUST NOT be used (excluding documentation).

902 [R 61] `xsd:substitutionGroup` MUST NOT be used.

903 [R 62] `xsd:ID/IDREF` MUST NOT be used.

904 [R 63] `xsd:key/xsd:keyref` MUST be used for information association.

905 [R 64] The absence of a construct or data MUST NOT carry meaning.

906 6.2 Attribute and Element Declarations

907 6.2.1 Attributes

908 6.2.1.1 Usage of Attributes

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

911 [R 65] User declared attributes MUST only be used to convey core component type (CCT)
912 supplementary component information.

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

915 [R 66] An attribute of a supplementary component with variable information MUST be based on the
916 appropriate built-in XSD data type.

917 [R 67] An attribute of a supplementary component which represents codes MUST be based on the
918 `xsd:simpleType` of the appropriate code list

919 [R 68] An attribute of a supplementary component which represents identifiers MUST be based on
920 the `xsd:simpleType` of the appropriate identifier scheme.

921 6.2.1.2 Constraints on Attribute Declarations

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

928 [R 69] The `xsd:nilable` attribute MUST NOT be used.

929 6.2.2 Elements

930 6.2.2.1 Usage of Elements

931 Elements are declared for document level message assembly, BBIEs, and ASBIEs.

932 6.2.2.2 Element Declaration

933 [R 70] All element declarations MUST be local except for a root element that must be declared
934 globally.

935 [R 71] Empty elements MUST NOT be used.

936 The `xsd:enumeration` element may be used within reusable or internal schema modules if the list of
937 enumerated values is less than 10, are not represented by a token, and are considered by TBG to be
938 static and particular to the business processes.

939 [R 72] The `xsd:type` of each leaf element declaration MUST be of the data type of its source
940 business information entity (BBIE) or complex type of its source association business
941 information entity (ASBIE).

942 Example 6-2:

```
943 <xsd:complexType name="AccountType">  
944   <xsd:annotation>  
945     ...see annotation...  
946   </xsd:annotation>  
947   <xsd:sequence>  
948     <xsd:element name="ID" type="udt:IdentifierType"  
949       minOccurs="0" maxOccurs="unbounded">  
950       <xsd:annotation>  
951         ...see annotation...  
952       </xsd:annotation>  
953     </xsd:element>  
954     <xsd:element name="Status" type="ram:StatusType"  
955       minOccurs="0" maxOccurs="unbounded">  
956       <xsd:annotation>  
957         ...see annotation...  
958       </xsd:annotation>  
959     </xsd:element>  
960     <xsd:element name="Name" type="udt:NameType"  
961       minOccurs="0" maxOccurs="unbounded">  
962       <xsd:annotation>  
963         ...see annotation...  
964       </xsd:annotation>  
965     </xsd:element>  
966   </xsd:sequence>  
967 </xsd:complexType>
```

968 6.2.2.3 Constraints on Element Declarations

969 [R 73] The `xsd:a11` element MUST NOT be used.

970 6.3 Type Definitions

971 6.3.1 Usage of Types

972 [R 74] All type definitions MUST be named.

973 Example 6-3:

```
974 <xsd:complexType name="AccountType">
975   <xsd:annotation>
976     ... see annotation ...
977   </xsd:annotation>
978   <xsd:sequence>
979     ... see element declaration ...
980   </xsd:sequence>
981 </xsd:complexType>
```

982 [R 75] Data type definitions MUST NOT duplicate the functionality of an existing data type
983 definition..

984 6.3.2 Simple Type Definitions

985 Built-in simple types must always be used where they satisfy the business requirements. Where the
986 business requirements cannot be satisfied, user defined complex type definitions will be used.

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

```
988 <xsd:simpleType name="DateTimeType">
989   <xsd:annotation>
990     ... see annotation ...
991   </xsd:annotation>
992   <xsd:restriction base="xsd:dateTime"/>
993 </xsd:simpleType>
```

994 **Example 6-5: Simple Types in Code Lists Module**

```
995 <xsd:simpleType name="CurrencyCodeContentType">
996   <xsd:restriction base="xsd:token">
997     <xsd:enumeration value="ADP">
998       ...see enumeration of code lists ...
999     </xsd:enumeration>
1000   </xsd:restriction>
1001   <xsd:annotation>
1002     ... see annotation ...
1003   </xsd:annotation>
1004 </xsd:simpleType>
```

1005 6.3.3 Complex Type Definitions

1006 User defined complex types may be used when built-in simple types do not satisfy the business
1007 requirements or when an aggregate business information entity (ABIE) must be defined.

1008 **Example 6-6: Complex Type of Object Class "AccountType"**

```
1009 <xsd:complexType name="AccountType">
1010   <xsd:annotation>
1011     ... see annotation ...
1012   </xsd:annotation>
1013   <xsd:sequence>
1014     ... see element declaration ...
1015   </xsd:sequence>
1016 </xsd:complexType>
```

1017 **6.4 Use of XSD Extension and Restriction**

1018 The general philosophy is that all UN/CEFACT XSD schema constructs will follow the model defined in
1019 Figure 5.1. These schema constructs are based on the concept that the underlying semantic structures
1020 of the core components and business information entities are normative forms of standards that
1021 developers are not allowed to alter without coordination of appropriate TBG groups (including TBG17 -
1022 Harmonization) and ICG. Accordingly, as business requirements dictate, new schema constructs will be
1023 created and new types defined and elements declared as appropriate. The concept of derivation through
1024 the use of `xsd:extension` and `xsd:restriction` will only be used in limited circumstances as
1025 described below.

1026 **6.4.1 Extension**

1027 [R 76] `xsd:extension` MUST only be used in the `cct:CoreComponentType` schema module and
1028 the `udt:UnqualifiedDataType` schema module. When used it MUST only extend a built-
1029 in XSD datatype.

1030 **6.4.2 Restriction**

1031 The CCTS specification employs the concept of semantic restriction in creating specific instantiations of
1032 core components. Accordingly, `xsd:restriction` will be used as appropriate to define types that are
1033 derived from the existing types. Where used, the derived types must always be renamed. Simple and
1034 complex type restrictions may be used.

1035 [R 77] When `xsd:restriction` is applied to a `xsd:simpleType` or `xsd:complexType` the
1036 derived construct MUST use a different name.

1037 **Example 6-7: Restriction of Simple Type**

```
1038 <xsd:simpleType name="IndicatorType">  
1039   <xsd:annotation>  
1040     ... see annotation ...  
1041   </xsd:annotation>  
1042   <xsd:restriction base="xsd:boolean">  
1043     <xsd:pattern value="false"/>  
1044     <xsd:pattern value="true"/>  
1045   </xsd:restriction>  
1046 </xsd:simpleType>
```

1047 **6.5 Annotation**

1048 All UN/CEFACT XSD schema constructs will use `xsd:annotation` to provide the documentation
1049 specified in Section 7 of CCTS.

1050 [R 78] Each UN/CEFACT defined or declared construct MUST use the `xsd:annotation` element
1051 for required CCTS documentation.

1052 [Note]

1053 In order to conform to this specification, this rule also applies to any construct imported
1054 from other standards bodies.

1055 **6.5.1 Documentation**

1056 The annotation documentation will be used to convey all metadata as specified in the CCTS, i.e., to
1057 convey the semantic content carried in the XML construct. The following annotations are required as
1058 defined in section 7 in type definitions and element declarations (the representation of each item in XML
1059 code is shown in parenthesis):

- 1060
- **Unique Identifier:** The unique identifier assigned to the artefact in the library. (UniqueID)
 - **Category Code:** The category to which the artefact belongs. (CategoryCode)
- 1061

- 1062 • **Dictionary Entry Name:** The complete name (not the tag name) of the artefact in the library.
1063 (DictionaryEntryName)
- 1064 • **Name:** The name of the message assembly. (Name)
- 1065 • **Version:** The version of the artefact as assigned by the registry. (VersionID)
- 1066 • **Definition:** The semantic meaning of the artefact. (Definition)
- 1067 • **Description:** A brief description of the business information exchange. (Description)
- 1068 • **Cardinality:** An indication of whether the property represents a not-applicable, optional,
1069 mandatory and/or repetitive characteristic of the object. (Cardinality)
- 1070 • **Business Domain:** The TBG groups(s) that developed the artefact. (BusinessDomain)
- 1071 • **Object Class Term:** The Object Class represented by the artefact. (ObjectClassTermName)
- 1072 • **Object Class Qualifier Term:** A term(s) that qualifies the Object Class..
1073 (ObjectClassQualifierTermName)
- 1074 • **Property Term:** The Property Term represented by the artefact. (PropertyTermName)
- 1075 • **Property Qualifier Term:** A term(s) that qualifies the Property Term.
1076 (PropertyQualifierTermName)
- 1077 • **Associated Object Class Term:** The Associated Object Class Term represented by the artefact.
1078 (AssociatedObjectClassTermName)
- 1079 • **Associated Object Class Qualifier Term:** A term(s) that qualifies the Associated Object Class
1080 Term. (AssociatedObjectClassQualifierTermName)
- 1081 • **Representation Term:** The Representation Term represented by the artefact.
1082 (RepresentationTermName)
- 1083 • **Data Type Qualifier Term:** A term(s) that qualifies the Data Type Term.
1084 (DataTypeQualifierTermName)
- 1085 • **Primitive Type:** The primitive data type as assigned to the artefact by CCTS. (PrimitiveType)
- 1086 • **Built In Type:** The XSD built-in data type assigned to the artefact. (BuiltInType)
- 1087 • **Business Process Context:** A valid value describing the Business Process contexts for which
1088 this construct has been designed. Default is "In All Contexts". (BusinessProcessContext)
- 1089 • **Geopolitical/Region Context:** A valid value describing the Geopolitical/Region contexts for
1090 which this construct has been designed. Default is "In All Contexts".
1091 (GeopoliticalOrRegionContext)
- 1092 • **Official Constraints Context:** A valid value describing the Official Constraints contexts for which
1093 this construct has been designed. Default is "None". (OfficialConstraintContext)
- 1094 • **Product Context:** A valid value describing the Product contexts for which this construct has been
1095 designed. Default is "In All Contexts". (ProductContext)
- 1096 • **Industry Context:** A valid value describing the Industry contexts for which this construct has
1097 been designed. Default is "In All Contexts". (IndustryContext)
- 1098 • **Business Process Role Context:** A valid value describing the Role contexts for which this
1099 construct has been designed. Default is "In All Contexts". (BusinessProcessRoleContext)
- 1100 • **Supporting Role Context:** A valid value describing the Supporting Role contexts for which this
1101 construct has been designed. Default is "In All Contexts". (SupportingRoleContext)
- 1102 • **System Capabilities Context:** A valid value describing the Systems Capabilities contexts for
1103 which this construct has been designed. Default is "In All Contexts". (SystemCapabilitiesContext)
- 1104 • **Usage Rule:** A constraint that describes specific conditions which are applicable to the artefact.
1105 (UsageRuleText)

1106 • **Business Term:** A synonym term under which the artefact is commonly known and used in
1107 business. (BusinessTermName)

1108 • **Example:** A possible value for the artefact. (Example)

1109 Appendix F specifies normative information on the specific annotation required for each of the artefacts.

1110 **Example 6-8: Example of annotation**

```
1111 <xsd:annotation>  
1112   <xsd:documentation xml:lang="en">  
1113     <ccts:UniqueID>UN00000002</ccts:UniqueID>  
1114     <ccts:CategoryCode>BBIE</ccts:CategoryCode>  
1115     <ccts:DictionaryEntryName>Account.  
1116       Identifier</ccts:DictionaryEntryName>  
1117     <ccts:Definition>The identification of a specific  
1118       account.</ccts:Definition>  
1119     <ccts:VersionID>1.0</ccts:VersionID>  
1120     <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>  
1121     <ccts:PropertyTermName>Identifier</ccts:PropertyTermName>  
1122     <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>  
1123     <ccts:BusinessTermName>Account Number</ccts:BusinessTermName>  
1124   </xsd:documentation>  
1125 </xsd:annotation>
```

1126 Each UN/CEFACT construct containing a code should include documentation that will identify the code
1127 list(s) that must be minimally supported when the construct is used.

1128 The following table provides a summary view of the annotation data as defined in section 6.

		<i>ism:RootSchema</i>	<i>ABIE xsd:complexType</i>	<i>BBIE xsd:element</i>	<i>ASBIE: xsd:element</i>	<i>cci:CoreComponentType</i>	<i>supplementary component</i>	<i>udt:UnqualifiedDataType</i>	<i>qdt:QualifiedDataType</i>
Unique Identifier	UniqueID	M	M	M	M	M	M	M	M
Category Code	CategoryCode	M	M	M	M	M	M	M	M
Dictionary Entry Name	DictionaryEntryName		M	M	M	M	M	M	M
Name	Name	M							
Version	VersionID	M	M	M	M	M		M	M
Definition	Definition		M	M	M	M	M	M	M
Description	Description	M							
Cardinality	Cardinality			M	M				
Business Domain	BusinessDomain	M, R							
Object Class Term	ObjectClassTermName		M	M	M		M		
Object Class Qualifier Term	ObjectClassQualifierTermName		O	O	O				
Property Term	PropertyTermName			M	M		M		
Property Qualifier Term	PropertyQualifierTermName			O	O				
Associated Object Class Term	AssociatedObjectClassTermName				M				
Associated Object Class Qualifier Term	AssociatedObjectClassQualifierTermName				O				
Representation Term	RepresentationTermName			M		M	M	M	M
Data Type Qualifier Term	DataTypeQualifierTermName								M
Primitive Type	PrimitiveType					M	M	M	M
XSD Built-in data type	BuiltInType						C	M	M
Business Process Context	BusinessProcessContext	M, R	O, R	O, R	O, R				O, R
Geopolitical/Region Context	GeopoliticalOrRegionContext	O, R	O, R	O, R	O, R				O, R
Official Constraints Context	OfficialConstraintContext	O, R	O, R	O, R	O, R				O, R
Product Context	ProductContext	O, R	O, R	O, R	O, R				O, R
Industry Context	IndustryContext	O, R	O, R	O, R	O, R				O, R
Business Process Role Context	BusinessProcessRoleContext	O, R	O, R	O, R	O, R				O, R
Supporting Role Context	SupportingRoleContext	O, R	O, R	O, R	O, R				O, R
System Capabilities Context	SystemCapabilitiesContext	O, R	O, R	O, R	O, R				O, R
Usage Rule	UsageRuleText		O, R	O, R	O, R	O, R	O, R	O, R	O, R
Business Term	BusinessTermName		O, R	O, R	O, R	O, R	O, R	O, R	O, R
Example	Example		O, R	O, R	O, R	O, R	O, R	O, R	O, R

1129 Key:

1130 M - mandatory

1131 O - optional

1132 R - repeating

1133 C - conditional

1134 7 XML Schema Modules

1135 This section describes the requirements of the various XML schema modules that will be incorporated
1136 within the UN/CEFACT library.

1137 7.1 Root Schema

1138 The root schema serves as the container for all other schema content that is required to fulfil a business
1139 information exchange. The root schema resides in its own namespace and imports external schema
1140 modules as needed. It may also include internal schema modules that reside in its namespace.

1141 7.1.1 Schema Construct

1142 Each root schema will be constructed in a standardized format in order to ensure consistency and ease of
1143 use. The specific format is shown in the example below and must adhere to the format of the relevant
1144 sections as detailed in Appendix B.

1145 Example 7-1: Structure of RootSchema Module

```
1146 <?xml version="1.0" encoding="UTF-8"?>
1147 <!-- ===== -->
1148 <!-- ===== [MODULENAME] Schema Module; [VERSION] ===== -->
1149 <!-- ===== -->
1150 <!--
1151     Module of      [MODULENAME]
1152     Agency:       UN/CEFACT
1153     Version:      0.3 Rev. 6
1154     Last change:  25 June 2004
1155
1156     Copyright (C) UN/CEFACT (2004). All Rights Reserved.
1157
1158     ... see copyright information ...
1159 -->
1160 <xsd:schema
1161   targetNamespace="urn:un:unece:unefact:data:draft:[MODULENAME]:0.3.6"
1162   ... see namespaces ...
1163   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1164   elementFormDefault="qualified" attributeFormDefault="unqualified">
1165   <!-- ===== Includes ===== -->
1166   <!-- ===== -->
1167   <!-- ===== Include of [MODULENAME] ===== -->
1168   ... see includes ...
1169   <!-- ===== Imports ===== -->
1170   <!-- ===== -->
1171   <!-- ===== Import of [MODULENAME] ===== -->
1172   ... see imports ...
1173
1174   <!-- ===== Root Element ===== -->
1175   <!-- ===== -->
1176   ... see root element declaration ...
1177   <!-- ===== Type Definitions ===== -->
1178   <!-- ===== -->
1179   <!-- ===== Type Definition: [TYPE] ===== -->
1180   <!-- ===== -->
1181   <xsd:complexType name="[TYPENAME]">
1182     <xsd:restriction base="xsd:token">
1183       ... see type definition ...
1184     </xsd:restriction>
1185   </xsd:complexType>
1186 </xsd:schema>
```

1187 7.1.2 Namespace Scheme

1188 All root schemas published by UN/CEFACT will be assigned a unique token by ATG to represent the
1189 namespace prefix. This token will be prefixed by 'rsm'.

1190 [R 79] The root schema module MUST be represented by a unique token.

1191 Example 7-2: Structure of Root Schema Module

1192

`xmlns:rsm="urn:un:unece:unefact:data:draft:UNCEFACTExamplesSchemaModule:0.3.6"`

1193

[Note]
Throughout this specification, the token 'rsm' is used for the unique root schema token.

1194

1195

7.1.3 Imports and Includes

1196

[R 80] The `rsm:RootSchema` MUST import the following schema modules:

1197

– `ram:ReusableABIE` Schema Module

1198

– `udt:UnqualifiedDataType` Schema Module

1199

– `qdt:QualifiedDataType` Schema Module

1200

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

1201

1202

1203

1204

1205

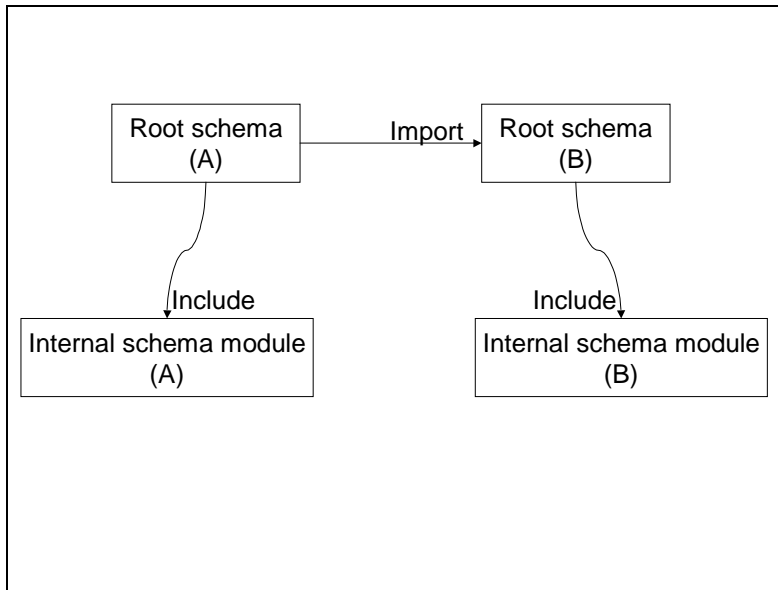
1206

1207

1208

1209

1210



1211

1212

[R 81] A `rsm:RootSchema` in one UN/CEFACT namespace that is dependent upon type definitions or element declaration defined in another namespace MUST import the `rsm:RootSchema` from that namespace.

1213

1214

1215

[R 82] A `rsm:RootSchema` in one UN/CEFACT namespace that is dependant upon type definitions or element declarations defined in another namespace MUST NOT import Schema Modules from that namespace other than the `rsm:RootSchema`.

1216

1217

1218

[R 83] The `rsm:RootSchema` MUST include any internal schema modules that reside in the root schema namespace.

1219

1220

7.1.4 Root Element Declaration

1221

Each UN/CEFACT business message has a single root element that is globally declared in the root schema.. The root element is named according to the business information exchange that it represents and references the message assembly that contains the actual business information.

1222

1223

1224 [R 84] A single global element known as the root element MUST be globally declared in a
1225 **rsm:RootSchema**.

1226 [R 85] The name of the root element MUST be the name of the Message Assembly with separators
1227 and spaces removed.

Example 7-3: Name of Root Element

```
1228 <!-- ===== Root Element ===== -->  
1229 <!-- ===== -->  
1230 <xsd:element name="PurchaseOrder" type="rsm:PurchaseOrderType">  
1231 <xsd:annotation>  
1232 ... see annotation ...  
1233 </xsd:annotation>  
1234 </xsd:element>  
1235
```

7.1.5 Type Definitions

1236 Root schemas are limited to defining a single **xsd:complexType** and a declaring a single global
1237 element that fully describe the business information exchange.

1239 [R 86] Root schema MUST define a single **xsd:complexType** that fully describes the business
1240 information exchange.

1241 [R 87] The name of the top-level complex type MUST be the name of the root element with the word
1242 "Type" appended.

1243 [R 88] The **xsd:complexType** of the root element must be the top-level complex type.

Example 7-4: Name of Complex Type Definition

```
1244 <!-- ===== Root Element ===== -->  
1245 <!-- ===== -->  
1246 <xsd:element name="PurchaseOrder" type="rsm:PurchaseOrderType">  
1247 <xsd:annotation>  
1248 ... see annotation ...  
1249 </xsd:annotation>  
1250 <xsd:complexType name="PurchaseOrderType">  
1251 <xsd:sequence>  
1252 ...  
1253 </xsd:sequence>  
1254 </xsd:complexType>  
1255 </xsd:element>  
1256
```

7.1.6 Annotations

1258 [R 89] For every **rsm:RootSchema** root element declaration a structured set of annotations MUST
1259 be present in the following pattern:

- 1260 • UniqueID (mandatory): The identifier that references the Message Assembly instance in a
1261 unique and unambiguous way.
- 1262 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
1263 will always be RSM.
- 1264 • Name (mandatory): The name of the Message Assembly
- 1265 • VersionID (mandatory): An indication of the evolution over time of a Message Assembly.
- 1266 • Description (mandatory): A brief description of the business information exchange.
- 1267 • BusinessDomain (mandatory, repetitive): The TBG group(s) that developed this Message
1268 Assembly.
- 1269 • BusinessProcessContext (mandatory, repetitive): The business process with which this
1270 Message Assembly is associated.
- 1271 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this
1272 Message Assembly.

1273
1274
1275
1276
1277
1278
1279
1280
1281
1282

- OfficialConstraintContext (optional, repetitive): The official constraint context for this Message Assembly.
- ProductContext (optional, repetitive): The product context for this Message Assembly.
- IndustryContext (optional, repetitive): The industry context for this Message Assembly.
- BusinessProcessRoleContext (optional, repetitive): The role context for this Message Assembly.
- SupportingRoleContext (optional, repetitive): The supporting role context for this Message Assembly.
- SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this Message Assembly.

1283
1284
1285
1286
1287

7.2 Internal Schema

A UN/CEFACT internal schema module will contain schema constructs representing ABIEs that are specific to a given root schema. Internal schema modules reside in the same namespace as their root schema. These constructs are subject to the same rules as those for reusable ABIEs as provided in sections 7.3.4, 7.3.5, and 7.3.6.

1288

7.2.1 Schema Construct

1289
1290
1291

Each internal schema will be constructed in a standardized format in order to ensure consistency and ease of use. Each internal schema format must adhere to the format of the relevant sections as detailed in Appendix B.

1292

7.2.2 Namespace Scheme

1293
1294

[R 90] All UN/CEFACT internal schema modules MUST be in the same namespace as their corresponding `rsm:RootSchema`.

1295
1296
1297

The UN/CEFACT internal schema modules do not declare a target namespace, but instead reside in the namespace of their parent root schema. All internal schema modules are accessed from the root schema using `xsd:include`.

1298
1299

[R 91] The internal schema module MUST be represented by the same token as its `rsm:RootSchema`.

1300

7.2.3 Imports and Includes

1301
1302

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

1303

7.3 Reusable Aggregate Business Information Entities

1304
1305
1306

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.

1307

7.3.1 Schema Construct

1308
1309
1310

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.

1311

Example 7-5: Structure of Reusable ABIEs Schema Module

1312
1313
1314
1315
1316

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ===== -->
<!-- ===== RAM Reusable ABIEs Schema Module ===== -->
<!-- ===== -->
<!--
```

```

1317 Module of Reusable ABIEs (Aggregate Business Information Entities)
1318 Agency: UN/CEFACT
1319 Version: 0.3 Rev. 6
1320 Last change: 25 June 2004
1321
1322 Copyright (C) UN/CEFACT (2004). All Rights Reserved.
1323
1324 ... see copyright information ...
1325 -->
1326 <xsd:schema
1327 targetNamespace=
1328 ... see namespace declaration ...
1329 xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
1330 attributeFormDefault="unqualified">
1331 <!-- ===== Imports ===== -->
1332 ... see imports ...
1333 <!-- ===== Type Definitions ===== -->
1334 <!-- =====
1335 ... see type defintions ...
1336 </xsd:schema>

```

1337 7.3.2 Namespace Scheme

1338 [R 92] The Reusable Aggregate Business Information Entity schema module MUST be represented
1339 by the token "ram".

1340 Example 7-6: Namespace of Reusable Aggregate Business Information Entity Schema Module

```

1341 "urn:un:unece:uncefact:data:draft:UNCEFACTReusableAggregateBusinessInformationEntitySc
1342 hemaModule:0.3.6"

```

1343 Example 7-7: Schema-Element of Reusable ABIEs Schema Module

```

1344 <xsd:schema
1345 targetNamespace=
1346 "urn:un:unece:uncefact:data:draft:UNCEFACTReusableAggregateBusinessInformationEntitySc
1347 hemaModule:0.3.6"
1348 xmlns:ram=
1349 "urn:un:unece:uncefact:data:draft:UNCEFACTReusableAggregateBusinessInformationSchemaMo
1350 dule:0.3.6"

```

1351 7.3.3 Imports and Includes

1352 [R 93] The `ram:ReusableAggregateBusinessInformationEntity` schema MUST import the
1353 following schema modules:
1354 - `udt:UnqualifiedDataType` Schema Module
1355 - `qdt:QualifiedDataType` Schema Module

1356 Example 7-8: Import of required modules

```

1357 <!-- ===== Imports ===== -->
1358 <!-- =====
1359 <!-- ===== Import of Qualified Data Type Schema Module (QDT) ===== -->
1360 <!-- =====
1361 <xsd:import
1362 namespace=
1363 "urn:un:unece:uncefact:data:draft:UNCEFACTQualifiedDataTypeSchemaModule:0.3.6"
1364 schemaLocation="http://www.unece.org/uncefact/data/
1365 UNCEFACTQualifiedDataTypeSchemaModule_0.3.6_draft.xsd"/>
1366 <!-- =====
1367 <!-- ===== Import of Unqualified Data Type Schema Module (UDT) ===== -->
1368 <!-- =====
1369 <xsd:import
1370 namespace=
1371 "urn:un:unece:uncefact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:
1372 0.3.6"
1373
1374 schemaLocation="http://www.unece.org/uncefact/data/UNCEFACTUnqualifiedDataTypesSchemaM
1375 odule_0.3.6_draft.xsd"/>

```


1376 7.3.4 Type Definitions

1377 [R 94] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named
1378 **xsd:complexType** MUST be defined.

1379 [R 95] The name of the ABIE **xsd:complexType** MUST be the **ccts:DictionaryEntryName**
1380 with the separators removed and with the "Details" suffix replaced with "Type".

1381 For every complex type definition based on an ABIE object class, its **xsd:content** model will be defined
1382 such that it reflects each property of the object class as a local element declaration, with its cardinality
1383 and sequencing within the schema XSD **content** model determined by the details of the source
1384 business information entity (ABIE).

1385 [R 96] Every aggregate business information entity (ABIE) **xsd:complexType** definition
1386 **xsd:content** model MUST use the **xsd:sequence** and/or **xsd:choice** elements with
1387 appropriate local element declarations to reflect each property (BBIE or ASBIE) of its class.

1388 [R 97] Recursion of **xsd:sequence** and/or **xsd:choice** MUST NOT occur.

1389 No complex type may contain a sequence followed by another sequence or a choice followed by another
1390 choice. However, it is permissible to alternate sequence and choice as in example 7.9.

1391 Example 7-9: Sequence within an object class

```
1392 <xsd:complexType name="AccountType" >  
1393   <xsd:annotation>  
1394     ...see annotation...  
1395   </xsd:annotation>  
1396   <xsd:sequence>  
1397     <xsd:element name="ID" type="udt:IdentifierType"  
1398       minOccurs="0" maxOccurs="unbounded">  
1399       <xsd:annotation>  
1400         ...see annotation...  
1401       </xsd:annotation>  
1402     </xsd:element>  
1403     <xsd:element name="Status" type="ram:StatusType"  
1404       minOccurs="0" maxOccurs="unbounded">  
1405       <xsd:annotation>  
1406         ...see annotation...  
1407       </xsd:annotation>  
1408     </xsd:element>  
1409     <xsd:element name="Name" type="udt:NameType"  
1410       minOccurs="0" maxOccurs="unbounded">  
1411       <xsd:annotation>  
1412         ...see annotation...  
1413       </xsd:annotation>  
1414     </xsd:element>  
1415     ...  
1416   </xsd:sequence>  
1417 </xsd:complexType>
```

1418 Example 7-10: Choice

```
1419 <xsd:complexType name="LocationType">  
1420   <xsd:annotation>  
1421     ... see annotation ...  
1422   </xsd:annotation>  
1423   <xsd:choice>  
1424     <xsd:element name="GeoCoordinate" type="ram:GeoCoordinateType"  
1425       minOccurs="0">  
1426       <xsd:annotation>  
1427         ... see annotation ...  
1428       </xsd:annotation>  
1429     </xsd:element>  
1430     <xsd:element name="Address" type="ram:AddressType"  
1431       minOccurs="0">  
1432       <xsd:annotation>  
1433         ... see annotation ...  
1434       </xsd:annotation>  
1435     </xsd:element>  
1436     <xsd:element name="Location" type="ram:LocationType"  
1437       minOccurs="0">  
1438       <xsd:annotation>
```

```

1439     ... see annotation ...
1440     </xsd:annotation>
1441   </xsd:element>
1442 </xsd:choice>
1443 </xsd:complexType>

```

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

```

1444 <xsd:complexType name="PeriodType">
1445   ...
1446   <xsd:sequence>
1447     <xsd:element name="DurationDateTime"
1448       type="qdt:DurationDateTimeType" minOccurs="0"
1449       maxOccurs="unbounded">
1450       ...
1451     </xsd:element>
1452     ...
1453   </xsd:sequence>
1454   <xsd:choice>
1455     <xsd:sequence>
1456       <xsd:element name="StartTime" type="udt:TimeType"
1457         minOccurs="0">
1458         ...
1459       </xsd:element>
1460       <xsd:element name="EndTime" type="udt:TimeType"
1461         minOccurs="0">
1462         ...
1463       </xsd:element>
1464     </xsd:sequence>
1465     <xsd:sequence>
1466       <xsd:element name="StartDate" type="udt:DateType"
1467         minOccurs="0">
1468         ...
1469       </xsd:element>
1470       <xsd:element name="EndDate" type="udt:DateType"
1471         minOccurs="0">
1472         ...
1473       </xsd:element>
1474     </xsd:sequence>
1475     <xsd:sequence>
1476       <xsd:element name="StartDateTime" type="udt:DateTimeType"
1477         minOccurs="0">
1478         ...
1479       </xsd:element>
1480       <xsd:element name="EndDateTime" type="udt:DateTimeType"
1481         minOccurs="0">
1482         ...
1483       </xsd:element>
1484     </xsd:sequence>
1485   </xsd:choice>
1486 </xsd:sequence>
1487 </xsd:complexType>

```

[R 98] 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.

Example 7-12: Type definition of an ABIE

```

1491 <!-- ===== Type Definitions ===== -->
1492 <!-- ===== -->
1493 <xsd:complexType name="AccountType" >
1494   <xsd:annotation>
1495     ... see annotation ...
1496   </xsd:annotation>
1497   <xsd:sequence>
1498     <xsd:element name="ID" type="udt:IdentifierType"
1499       minOccurs="0" maxOccurs="unbounded">
1500       <xsd:annotation>
1501         ... see annotation ...
1502       </xsd:annotation>
1503     </xsd:element>
1504     ... see element declaration ....
1505   </xsd:sequence>
1506 </xsd:complexType>

```

1507 7.3.5 Element Declarations

- 1508 [R 99] For every attribute of an object class (BBIE) identified in an ABIE, a named `xsd:element`
1509 MUST be locally declared within the `xsd:complexType` representing that ABIE.
- 1510 [R 100] Each BBIE element name declaration MUST be based on the property term and qualifiers
1511 and the representation term of the basic business information entity (BBIE). If there are
1512 successive duplicate words in the property term and representation terms of the source
1513 dictionary entry name, then the duplicate words MUST be removed.
- 1514 [R 101] If the representation term of a BBIE is 'text', it MUST be removed.
- 1515 [R 102] The BBIE element MUST be based on an appropriate data type that is defined in the
1516 UN/CEFACT `qdt:QualifiedDataType` or `udt:UnqualifiedDataType` schema
1517 modules.
- 1518 [R 103] For every association (ASBIE) identified in the UN/CEFACT syntax-neutral model, a named
1519 `xsd:element` MUST be locally declared within the `xsd:complexType` representing the
1520 ABIE.
- 1521 [R 104] Each ASBIE element name declaration MUST be based on the property term and object
1522 class of the association business information entity (ASBIE). If there are successive duplicate
1523 words in the property term and object class of the associated ABIE, then the duplicate words
1524 MUST be removed.
- 1525 [R 105] The element representing an association business information entity (ASBIE) MUST be of the
1526 complex type corresponding to its associated aggregate business information (ABIE).

1527 Example 7-13: Element declaration within an ABIE

```
1528 ... see type definition ...  
1529 <xsd:element name="ID" type="udt:IdentifierType"  
1530 minOccurs="0" maxOccurs="unbounded">  
1531 <xsd:annotation>  
1532 ... see annotation ...  
1533 </xsd:annotation>  
1534 </xsd:element>  
1535 <xsd:element name="Status" type="ram:StatusType"  
1536 minOccurs="0" maxOccurs="unbounded">  
1537 <xsd:annotation>  
1538 ... see annotation ...  
1539 </xsd:annotation>  
1540 </xsd:element>  
1541 <xsd:element name="Name" type="udt:NameType"  
1542 minOccurs="0" maxOccurs="unbounded">  
1543 <xsd:annotation>  
1544 ... see annotation ...  
1545 </xsd:annotation>  
1546 </xsd:element>  
1547 <xsd:element name="CurrencyCode" type="qdt:CurrencyCodeType"  
1548 minOccurs="0" maxOccurs="unbounded">  
1549 <xsd:annotation>  
1550 ... see annotation ...  
1551 </xsd:annotation>  
1552 </xsd:element>  
1553 ... see type definition ...
```

1554 7.4 Annotation

- 1555 [R 106] For every ABIE `xsd:complexType` definition a structured set of annotations MUST be
1556 present in the following pattern:

- 1557 • UniqueID (mandatory): The identifier that references an ABIE instance in a unique and
1558 unambiguous way.
- 1559 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
1560 will always be ABIE.

- 1561 • DictionaryEntryName (mandatory): The official name of an ABIE.
- 1562 • VersionID (mandatory): An indication of the evolution over time of an ABIE instance.
- 1563 • Definition (mandatory): The semantic meaning of an ABIE.
- 1564 • ObjectClassTermName (mandatory): The Object Class Term of the ABIE.
- 1565 • QualifierTermName (optional): Qualifies the Object Class Term of the ABIE.
- 1566 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the ABIE.
- 1567
- 1568 • BusinessTermName (optional, repetitive): A synonym term under which the ABIE is commonly known and used in the business.
- 1569
- 1570 • BusinessProcessContext (optional, repetitive): The business process with which this ABIE is associated.
- 1571
- 1572 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this ABIE.
- 1573
- 1574 • OfficialConstraintContext (optional, repetitive): The official constraint context for this ABIE.
- 1575 • ProductContext (optional, repetitive): The product context for this ABIE.
- 1576 • IndustryContext (optional, repetitive): The industry context for this ABIE.
- 1577 • BusinessProcessRoleContext (optional, repetitive): The role context for this ABIE.
- 1578 • SupportingRoleContext (optional, repetitive): The supporting role context for this ABIE.
- 1579 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this ABIE.
- 1580
- 1581 • Example (optional, repetitive): Example of a possible value of an ABIE.

Example 7-14: Annotation of an ABIE

```

1582 <xsd:complexType name="AccountType" >
1583   <xsd:annotation>
1584     <xsd:documentation xml:lang="en">
1585       <ccts:UniqueID>UN00000001</ccts:UniqueID>
1586       <ccts:CategoryCode>ABIE</ccts:CategoryCode>
1587       <ccts:DictionaryEntryName>Account.
1588 Details</ccts:DictionaryEntryName>
1589       <ccts:VersionID>1.0</ccts:VersionID>
1590       <ccts:Definition> A business arrangement whereby debits and/or
1591 credits arising from transactions are recorded. This could be with a bank,
1592 i.e. a financial account, or a trading partner offering supplies or services
1593 'on account', i.e. a commercial account</ccts:Definition>
1594       <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
1595     </xsd:documentation>
1596   </xsd:annotation>
1597   ...
1598 </xsd:complexType>

```

[R 107] For every BBIE `xsd:element` declaration a structured set of annotations MUST be present in the following pattern:

- 1602 • UniqueID (mandatory): The identifier that references a BBIE instance in a unique and unambiguous way.
- 1603
- 1604 • CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be BBIE.
- 1605
- 1606 • Dictionary Entry Name (mandatory): The official name of the BBIE.
- 1607 • VersionID (mandatory): An indication of the evolution over time of a BBIE instance.
- 1608 • Definition (mandatory): The semantic meaning of the BBIE.

- 1609 • Cardinality (mandatory): Indication whether the BIE Property represents a not-applicable,
1610 optional, mandatory and/or repetitive characteristic of the ABIE.
- 1611 • ObjectClassTermName (mandatory): The Object Class Term of the parent ABIE.
- 1612 • ObjectClassQualifierTermName (optional): Qualifies the Object Class Term of the parent
1613 ABIE.
- 1614 • PropertyTermName (mandatory): The Property Term of the BBIE.
- 1615 • PropertyQualifierTermName (optional): Qualifies the Property Term of the BBIE.
- 1616 • RepresentationTermName (mandatory): The Representation Term of the BBIE.
- 1617 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
1618 applicable to the BBIE.
- 1619 • BusinessProcessContext (optional, repetitive): The business process with which this BBIE is
1620 associated.
- 1621 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this
1622 BBIE.
- 1623 • OfficialConstraintContext (optional, repetitive): The official constraint context for this BBIE.
- 1624 • ProductContext (optional, repetitive): The product context for this BBIE.
- 1625 • IndustryContext (optional, repetitive): The industry context for this BBIE.
- 1626 • BusinessProcessRoleContext (optional, repetitive): The role context for this BBIE.
- 1627 • SupportingRoleContext (optional, repetitive): The supporting role context for this BBIE.
- 1628 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
1629 BBIE.
- 1630 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
1631 applicable to this BBIE.
- 1632 • BusinessTermName (optional, repetitive): A synonym term under which the BBIE is
1633 commonly known and used in the business.
- 1634 • Example (optional, repetitive): Example of a possible value of a BBIE.

1635 **Example 7-15: Annotation of a BBIE**

```

1636 <xsd:element name="ID" type="udt:IdentifierType"
1637   minOccurs="0" maxOccurs="unbounded">
1638   <xsd:annotation>
1639     <xsd:documentation xml:lang="en">
1640       <ccts:UniqueID>UN00000002</ccts:UniqueID>
1641       <ccts:CategoryCode>BBIE</ccts:CategoryCode>
1642       <ccts:DictionaryEntryName>Account.
1643 Identifier</ccts:DictionaryEntryName>
1644       <ccts:VersionID>1.0</ccts:VersionID>
1645       <ccts:Definition>The identification of a specific account.
1646       </ccts:Definition>
1647       </ccts:Cardinality>0..n</ccts:Cardinality>
1648       <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
1649       <ccts:PropertyTermName>Identifier</ccts:PropertyTermName>
1650       <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
1651       <ccts:BusinessTermName>Account Number</ccts:BusinessTermName>
1652     </xsd:documentation>
1653   </xsd:annotation>
1654 </xsd:element>

```

1655 [R 108] For every ASBIE **xsd:element** declaration a structured set of annotations MUST be present
1656 in the following pattern:

- 1657 • UniqueID (mandatory): The identifier that references an ASBIE instance in a unique and
1658 unambiguous way.

- 1659 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
1660 will always be ASBIE.
- 1661 • DictionaryEntryName (mandatory): The official name of the ASBIE.
- 1662 • VersionID (mandatory): An indication of the evolution over time of the ASBIE instance.
- 1663 • Definition (mandatory): The semantic meaning of the ASBIE.
- 1664 • Cardinality (mandatory): Indication whether the ASBIE Property represents a not-applicable,
1665 optional, mandatory and/or repetitive characteristic of the ABIE.
- 1666 • ObjectClassTermName (mandatory): The Object Class Term of the associating ABIE.
- 1667 • ObjectClassQualifierTermName (Optional): A term that qualifies the Object Class Term of the
1668 associating ABIE.
- 1669 • PropertyTermName (mandatory): The Property Term of the ASBIE.
- 1670 • PropertyQualifierTermName (Optional): A term that qualifies the Property Term of the ASBIE.
- 1671 • AssociatedObjectClassTermName (mandatory): The Object Class Term of the associated
1672 ABIE.
- 1673 • AssociatedObjectClassQualifierTermName (optional): Qualifies the Object Class Term of the
1674 associated ABIE.
- 1675 • BusinessProcessContext (optional, repetitive): The business process with which this ASBIE
1676 is associated.
- 1677 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this
1678 ASBIE.
- 1679 • OfficialConstraintContext (optional, repetitive): The official constraint context for this ASBIE.
- 1680 • ProductContext (optional, repetitive): The product context for this ASBIE.
- 1681 • IndustryContext (optional, repetitive): The industry context for this ASBIE.
- 1682 • BusinessProcessRoleContext (optional, repetitive): The role context for this ASBIE.
- 1683 • SupportingRoleContext (optional, repetitive): The supporting role context for this ASBIE.
- 1684 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
1685 ASBIE.
- 1686 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
1687 applicable to the ASBIE.
- 1688 • BusinessTermName (optional, repetitive): A synonym term under which the ASBIE is
1689 commonly known and used in the business.
- 1690 • Example (optional, repetitive): Example of a possible value of an ASBIE.

1691 **Example 7-16: Annotation of an ASBIE**

```

1692 <xsd:element name="Status" type="ram:StatusType"
1693   minOccurs="0" maxOccurs="unbounded">
1694   <xsd:annotation>
1695     <xsd:documentation xml:lang="en">
1696       <ccts:UniqueID>UN00000003</ccts:UniqueID>
1697       <ccts:CategoryCode>ASCC</ccts:CategoryCode>
1698       <ccts:DictionaryEntryName>Account.
1699 Status</ccts:DictionaryEntryName>
1700       <ccts:VersionID>1.0</ccts:VersionID>
1701       <ccts:Definition>Associated status information related to
1702 account details.</ccts:Definition>
1703       <ccts:Cardinality>0..n</ccts:Cardinality>
1704       <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
1705       <ccts:PropertyTermName>Status</ccts:PropertyTermName>
1706       <ccts:AssociatedObjectClassTermName>Status
1707       </ccts:AssociatedObjectClassTermName>
1708     </xsd:documentation>

```

1709
1710

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

1711 7.5 Core Component Type

1712 7.5.1 Use of Core Component Type Module

1713 The purpose of the core component type module is to define the core component types on which the
1714 unqualified data types are based. This module is only for reference and will not be included/imported in
1715 any schema. The normative formatted schema for the Core Component Type module is contained in
1716 Appendix D.

1717 7.5.2 Schema Construct

1718 The core component type schema module will be constructed in a standardized format in order to ensure
1719 consistency and ease of use. The specific format is shown below and must adhere to the format of the
1720 relevant sections as detailed in Appendix B.

1721 Example 7-17: Structure of Core Component Type Schema Module

```
1722 <?xml version="1.0" encoding="utf-8"?>  
1723 <!-- ===== -->  
1724 <!-- ===== CCTS Core Component Type Schema Module ===== -->  
1725 <!-- ===== -->  
1726 <!--  
1727     Module of      Core Component Type  
1728     Agency:       UN/CEFACT  
1729     Version:      0.3 Rev. 6  
1730     Last change:  25 June 2004  
1731  
1732     Copyright (C) UN/CEFACT (2004). All Rights Reserved.  
1733  
1734     ... see copyright information ...  
1735  
1736 -->  
1737 <xsd:schema  
1738     targetNamespace=  
1739     ... see namespace ...  
1740     xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
1741     elementFormDefault="qualified" attributeFormDefault="unqualified">  
1742     <!-- ===== Type Definitions ===== -->  
1743     <!-- ===== -->  
1744     <!-- ===== CCT: AmountType ===== -->  
1745     <!-- ===== -->  
1746     ... see type definitions ...  
1747 </xsd:schema>
```

1748 7.5.3 Namespace Scheme

1749 [R 109] The core component type (CCT) schema module MUST be represented by the token "cct".

1750 Example 7-18: Namespace of Core Component Type Schema Module

```
1751 "urn:un:unece:uncefact:documentation:draft:UNCEFACTCCTSCCTSchemaModule:0.3.6"
```

1752 Example 7-19: Namespace of Core Component Type Schema Module

```
1753 <xsd:schema  
1754     targetNamespace=  
1755     "urn:un:unece:uncefact:documentation:draft:UNCEFACTCCTSCCTSchemaModule:0.3.6"  
1756     "  
1757     xmlns:cct=  
1758     "urn:un:unece:uncefact:documentation:draft:UNCEFACTCCTSCCTSchemaModule:0.3.6"  
1759     "  
1760     xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
1761     elementFormDefault="qualified" attributeFormDefault="unqualified">
```

1762 7.5.4 Imports and Includes

1763 The core component types schema module does not import or include any other schema modules.

1764 [R 110] The `cct:CoreCoreComponentType` schema module MUST NOT include or import any
1765 other schema modules.

1766 7.5.5 Type Definitions

1767 [R 111] Every `cct:CoreComponentType` MUST be defined as a named `xsd:complexType` in the
1768 `cct:CoreComponentType` schema module.

1769 [R 112] The name of each `xsd:complexType` based on a `cct:CoreComponentType` MUST be
1770 the dictionary entry name of the core component type (CCT), with the separators and spaces
1771 removed.

1772 [R 113] Each `cct:CoreComponentType` `xsd:complexType` definition MUST contain one
1773 `xsd:simpleContent` element.

1774 [R 114] The `cct:CoreComponentType` `xsd:complexType` definition `xsd:simpleContent`
1775 element MUST contain one `xsd:extension` element. This `xsd:extension` element must
1776 include an XSD based attribute that defines the specific built-in XSD data type required for
1777 the CCT content component.

1778 [R 115] Within the `cct:CoreComponentType` `xsd:extension` element a `xsd:attribute` MUST be
1779 declared for each supplementary component pertaining to that `cct:CoreComponentType`.

1780 Example 7-20: Type definition of a CCT

```
1781 <!-- ===== Type Definitions ===== -->  
1782 <!-- ===== CCT: AmountType ===== -->  
1783 <!-- ===== CCT: AmountType ===== -->  
1784 <!-- ===== CCT: AmountType ===== -->  
1785 <xsd:complexType name="AmountType">  
1786   <xsd:annotation>  
1787     ... see annotation ...  
1788   </xsd:annotation>  
1789   <xsd:simpleContent>  
1790     <xsd:extension base="xsd:decimal">  
1791       <xsd:attribute name="currencyID" type="xsd:token" use="optional"  
1792       <xsd:annotation>  
1793         ... see annotation ...  
1794       </xsd:annotation>  
1795     </xsd:attribute>  
1796     ... see attribute declaration ...  
1797   </xsd:extension>  
1798 </xsd:simpleContent>  
1799 </xsd:complexType>
```

1800 7.5.6 Attribute Declarations

1801 The current CCTS does not specify the components of the CCT supplementary component dictionary
1802 entry name. However, in order to ensure a standard approach to declaring the supplementary
1803 components as attributes, ATG has applied the naming concepts from ISO 11179, part 5. Specifically,
1804 ATG has defined the dictionary entry name as it is stated in CCTS in terms of object class, property term,
1805 and representation term. These components are identified in the annotation documentation for each
1806 supplementary component in the CCT schema module.

1807 [R 116] Each `cct:CoreComponentType` supplementary component `xsd:attribute` "name"
1808 MUST be the CCTS supplementary component dictionary entry name with the separators
1809 and spaces removed.

1810 [R 117] If the object class of the supplementary component dictionary entry name contains the name
1811 of the representation term of the parent CCT, the duplicated object class word or words
1812 MUST be removed from the supplementary component `xsd:attribute` name.

1813 [R 118] If the object class of the supplementary component dictionary entry name contains the term
1814 'identification', the term 'identification' MUST be removed from the supplementary component
1815 `xsd:attribute` name.

1816 [R 119] If the representation term of the supplementary component dictionary entry name is 'text', the
1817 representation term MUST be removed from the supplementary component
1818 **xsd:attribute** name.

1819 [R 120] The attribute representing as supplementary component MUST be based on the appropriate
1820 built-in XSD data type.

1821 **Example 7-22: Supplementary component other than code or identifier**

```
1822 <xsd:complexType name="BinaryObjectType">  
1823 ...  
1824 <xsd:simpleContent>  
1825 <xsd:extension base="xsd:base64Binary">  
1826 <xsd:attribute name="format" type="xsd:string" use="optional">  
1827 ...  
1828 </xsd:attribute>  
1829 ...  
1830 </xsd:extension>  
1831 </xsd:simpleContent>  
1832 </xsd:complexType>
```

1833 **7.5.7 Extension and Restriction**

1834 The core component type schema module is a generic module that will be restricted in qualified and
1835 unqualified data type schema modules.

1836 **7.5.8 Annotation**

1837 [R 121] For every **cct:CoreComponentType** **xsd:complexType** definition a structured set of
1838 annotations MUST be present in the following pattern:

- 1839 • UniqueID (mandatory): The identifier that references the Core Component Type instance in a
1840 unique and unambiguous way.
- 1841 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
1842 will always be CCT.
- 1843 • DictionaryEntryName (mandatory): The official name of a Core Component Type.
- 1844 • VersionID (mandatory): An indication of the evolution over time of a Core Component Type
1845 instance.
- 1846 • Definition (mandatory): The semantic meaning of a Core Component Type.
- 1847 • RepresentationTermName (mandatory): The primary representation term of the Core
1848 Component Type.
- 1849 • PrimitiveType (mandatory): The primitive data type of the Core Component Type.
- 1850 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
1851 applicable to the Core Component Type.
- 1852 • BusinessTermName (optional, repetitive): A synonym term under which the Core Component
1853 Type is commonly known and used in the business.
- 1854 • Example (optional, repetitive): Example of a possible value of a Core Component Type.

1855 **Example 7-21: Annotation of a CCT**

```
1856 ... see type definition ...  
1857 <xsd:annotation>  
1858 <xsd:documentation xml:lang="en">  
1859 <ccts:UniqueID>UNDT000001</ccts:UniqueID>  
1860 <ccts:CategoryCode>CCT</ccts:CategoryCode>  
1861 <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>  
1862 <ccts:VersionID>1.0</ccts:VersionID>  
1863 <ccts:Definition>A number of monetary units specified in a currency  
1864 where the unit of the currency is explicit or  
1865 implied.</ccts:Definition>  
1866 <ccts:RepresentationTermName>Amount</ccts:RepresentationTermName>  
1867 <ccts:PrimitiveType>decimal</ccts:PrimitiveType>
```

```

1868     </xsd:documentation>
1869     </xsd:annotation>
1870     ... see type definition ...

```

[R 122] For every supplementary component `xsd:attribute` declaration a structured set of annotations MUST be present in the following pattern:

- UniqueID (mandatory): The identifier that references a Supplementary Component instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. 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.
- ObjectClassTermName (mandatory): The Object Class of the Supplementary Component.
- PropertyTermName (mandatory): The Property Term of the Supplementary Component.
- RepresentationTermName (mandatory): The 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.
- Example (optional, repetitive): Example of a possible value of a Basic Core Component.

Example 7-22: Annotation of a supplementary component

```

1887     ... see attribute declaration ...
1888     <xsd:annotation>
1889         <xsd:documentation xml:lang="en">
1890             <ccts:UniqueID>UNDT000001-SC2</ccts:UniqueID>
1891             <ccts:CategoryCode>SC</ccts:CategoryCode>
1892             <ccts:DictionaryEntryName>Amount. Currency.
1893 Identifier</ccts:DictionaryEntryName>
1894             <ccts:Definition>The currency of the amount.</ccts:Definition>
1895             <ccts:ObjectClassTermName>Amount</ccts:ObjectClassTermName>
1896             <ccts:PropertyTermName>Currency</ccts:PropertyTermName>
1897             <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
1898             <ccts:PrimitiveType>string</ccts:PrimitiveType>
1899             <ccts:UsageRule>Reference UNECE Rec 9, using 3-letter
1900 alphabetic codes.</ccts:UsageRule>
1901         </xsd:documentation>
1902     </xsd:annotation>
1903     ... see attribute declaration ...

```

7.6 Unqualified Data Type

7.6.1 Use of Unqualified Data Type Module

The unqualified data type schema module will define data types for all primary and secondary representation terms as specified in the CCTS. All data types will be defined as `xsd:complexType` or `xsd:simpleType` and will only reflect restrictions as specified in CCTS and agreed upon industry best practices.

7.6.2 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 7-23: Structure of unqualified data type schema module

```

1915 <?xml version="1.0" encoding="utf-8"?>
1916 <!-- ===== -->
1917 <!-- ===== UDT Unqualified Data Type Schema Module ===== -->
1918

```

```

1919 <!-- ===== Imports ===== -->
1920 <!--
1921     Module of      Unqualified Data Type
1922     Agency:       UN/CEFACT
1923     Version:      0.3 Rev.6
1924     Last change:  25 June 2004
1925
1926     Copyright (C) UN/CEFACT (2004). All Rights Reserved.
1927
1928     ... see copyright information ...
1929
1930 -->
1931 <xsd:schema targetNamespace=
1932     ... see namespace ...
1933     xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
1934     attributeFormDefault="unqualified">
1935     <!-- ===== Imports ===== -->
1936     <!-- ===== Imports ===== -->
1937     ... see imports ...
1938     <!-- ===== Type Definitions ===== -->
1939     <!-- ===== Type Definitions ===== -->
1940     <!-- ===== Primary RT: Amount. Type ===== -->
1941     <!-- ===== Primary RT: Amount. Type ===== -->
1942     <!-- ===== Primary RT: Amount. Type ===== -->
1943     <xsd:complexType name="AmountType">
1944     ... see type definition ...
1945     </xsd:complexType>
1946     ...
1947 </xsd:schema>

```

1948 7.6.3 Namespace Scheme

1949 [R 123] The Unqualified Data Type schema module namespace MUST be represented by the token
1950 "udt".

1951 Example 7-24: Namespace of unqualified data type schema module

```

1952 "urn:un:unece:unefact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:0.3
1953 .6"

```

1954 Example 7-25: Schema-element of unqualified data type schema module

```

1955 <xsd:schema
1956     targetNamespace=
1957     "urn:un:unece:unefact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:0.
1958     3.6"
1959     xmlns:udt=
1960     "urn:un:unece:unefact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:0.
1961     3.6"
1962     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1963     elementFormDefault="qualified" attributeFormDefault="unqualified">

```

1964 7.6.4 Imports and Includes

1965 The Unqualified Data Type schema will import the required code list and identifier list schema modules.

1966 [R 124] The `udt:UnqualifiedDataType` schema MUST NOT import any other schema modules
1967 than the following:

- 1968 - `ids:IdentifierList` schema modules
- 1969 - `clm:CodeList` schema modules

1970 Example 7-26: Imports

```

1971 <!-- ===== Imports ===== -->
1972 <!-- ===== Imports ===== -->
1973 <!-- ===== Imports of Code Lists ===== -->
1974 <!-- ===== Imports of Code Lists ===== -->
1975 <xsd:import namespace=
1976     "urn:un:unece:unefact:codelist:draft:6:3403:D.04A"
1977     schemaLocation="http://www.unece.org/unefact/codelist/63403_D.04A_draft.xsd
1978     "/>
1979 <!-- ===== Imports of Identifier Lists ===== -->
1980 <!-- ===== Imports of Identifier Lists ===== -->

```

1981
1982
1983
1984

```
<xsd:import namespace=  
  "urn:un:unece:unefact:identifierlist:draft:5:3166-1:1977"  
  schemaLocation="http://www.unece.org/unefact/identifierlist/53166-  
1.1997_standard.xsd"/>
```

1985

7.6.5 Type Definitions

1986
1987
1988

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

1989
1990
1991

[R 125] A **udt:UnqualifiedDataType** MUST be defined for each approved primary and secondary representation terms identified in the CCTS Permissible Representation Terms table.

1992
1993
1994

[R 126] The name of each **udt:UnqualifiedDataType** MUST be the dictionary entry name of the primary or secondary representation term, with "Type" at the end and the separators and spaces removed.

1995
1996
1997

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

1998
1999
2000
2001

[R 127] For every **udt:UnqualifiedDataType** whose supplementary components map directly to the properties of a built-in **xsd:dataTtpe**, the **udt:UnqualifiedDataType** MUST be defined as a named **xsd:simpleType** in the **udt:UnqualifiedDataType** schema module.

2002
2003
2004
2005

[R 128] Every **udt:UnqualifiedDataType** defined as a **xsd:simpleType** MUST contain one **xsd:restriction** element. This **xsd:restriction** element MUST include an **xsd:base** attribute that defines the specific built-in XSD data type required for the content component.

2006
2007

When the unqualified data type does not directly map to an **xsd:simpleType** due to the supplementary components needing to be expressed, it will be defined as an **xsd:complexType**.

2008
2009
2010
2011

[R 129] For every **udt:UnqualifiedDataType** whose supplementary components are not equivalent to the properties of a built-in XSD data type, a **udt:UnqualifiedDataType** MUST be defined as an **xsd:complexType** in the **udt:UnqualifiedDataType** schema module.

2012
2013

[R 130] Every **udt:UnqualifiedDataType xsd:complexType** definition MUST contain one **xsd:simpleContent** element.

2014
2015
2016
2017

[R 131] Every **udt:UnqualifiedDataType xsd:complexType xsd:simpleContent** element MUST contain one **xsd:extension** element. This **xsd:extension** element must include an **xsd:base** attribute that defines the specific built-in XSD datatype required for the content component.

2018

7.6.6 Attribute Declarations

2019
2020
2021
2022
2023

Each core component supplementary component will normally be declared as an attribute of the complex type. However, the namespace scheme for code lists and identification scheme lists has been designed to include some of the supplementary components for the CCTs Code. Type and Identifier. Type. Thus, those attributes that are included in the namespace will not be declared as part of the unqualified data type.

2024
2025
2026

[R 132] Within the **udt:UnqualifiedDataType xsd:complextype xsd:extension** element an **xsd:attribute** MUST be declared for each supplementary component pertaining to the underlying CCT, unless the attribute is contained in the namespace declaration.

2027 The attributes representing supplementary components will be named based on their underlying CCT
2028 supplementary component. The user declared attributes can be based on:

- 2029 • XSD built-in types, if a specific supplementary component represents a variable value
- 2030 • simpleTypes of a code list, if the specific supplementary component represents a code value
- 2031 • simpleTypes of an identifier scheme, if the specific supplementary component represents an
2032 identifier value.

2033 For some CCTs, the CCTS identifies restrictions in the form of pointing to certain restrictive code or
2034 identifier lists. These restrictive lists will be declared in the code list or identifier schema module and the
2035 unqualified data type will reference these.

2036 [R 133] Each supplementary component `xsd:attribute` name MUST be the supplementary
2037 component name with the separators and spaces removed.

2038 [R 134] If the object class of the supplementary component dictionary entry name contains the name
2039 of the representation term of the parent CCT, the duplicated object class word or words
2040 MUST be removed from the supplementary component `xsd:attribute` name.

2041 [R 135] If the object class of the supplementary component dictionary entry name contains the term
2042 'identification', the term 'identification' MUST be removed from the supplementary component
2043 `xsd:attribute` name.

2044 [R 136] If the representation term of the supplementary component dictionary entry name is 'text', the
2045 representation term MUST be removed from the supplementary component
2046 `xsd:attribute` name.

2047 Example 7-27: Type definitions of unqualified data types

```
2048 <!-- ===== Type Definitions ===== -->
2049 <!-- =====
2050 <!-- ===== Primary RT: Amount. Type ===== -->
2051 <!-- =====
2052 <xsd:complexType name="AmountType">
2053   <xsd:annotation>
2054     ... see annotation ...
2055   </xsd:annotation>
2056   <xsd:simpleContent>
2057     <xsd:extension base="xsd:decimal">
2058       <xsd:attribute name="currencyID"
2059         type="clm54217:CurrencyCodeContentType" use="required">
2060         <xsd:annotation>
2061           ... see annotation ...
2062         </xsd:annotation>
2063       </xsd:attribute>
2064       <xsd:attribute name="currencyCodeListVersionID"
2065         type="xsd:normalizedString" use="optional">
2066         <xsd:annotation>
2067           ... see annotation ...
2068         </xsd:annotation>
2069       </xsd:attribute>
2070     </xsd:extension>
2071   </xsd:simpleContent>
2072 </xsd:complexType>
2073 <!-- ===== Primary RT: Binary Object. Type ===== -->
2074 <!-- =====
2075 <xsd:complexType name="BinaryObjectType">
2076   <xsd:annotation>
2077     ... see annotation ...
2078   </xsd:annotation>
2079   <xsd:simpleContent>
2080     <xsd:extension base="xsd:base64Binary">
2081       <xsd:attribute name="mimeType"
2082         type="clmIANAMIMEMediaType:BinaryObjectMimeTypeContentType">
2083       <xsd:annotation>
2084         ... see annotation ...
2085       </xsd:annotation>
2086     </xsd:extension>
2087   </xsd:simpleContent>
2088 </xsd:complexType>
```

```

2087         </xsd:attribute>
2088         <xsd:attribute name="encodingCode" type="xsd:normalizedString"
2089 use="optional">
2090             <xsd:annotation>
2091                 ... see annotation ...
2092             </xsd:annotation>
2093         </xsd:attribute>
2094         <xsd:attribute name="characterSetCode" type="xsd:normalizedString"
2095 use="optional">
2096             <xsd:annotation>
2097                 ... see annotation ...
2098             </xsd:annotation>
2099         </xsd:attribute>
2100         <xsd:attribute name="uri" type="xsd:anyURI" use="optional">
2101             <xsd:annotation>
2102                 ... see annotation ...
2103             </xsd:annotation>
2104         </xsd:attribute>
2105         <xsd:attribute name="filename" type="xsd:string" use="optional">
2106             <xsd:annotation>
2107                 ... see annotation ...
2108             </xsd:annotation>
2109         </xsd:attribute>
2110     </xsd:extension>
2111 </xsd:simpleContent>
2112 </xsd:complexType>

```

2113 The user declared attributes are dependent on the type of representation term of the specific
2114 supplementary component. See Appendix G for the mapping of the representation terms to the user
2115 defined attributes.

2116 [R 137] If the representation term of the relevant supplementary component is a “Code” and
2117 validation is required, then the attribute representing this supplementary component **MUST**
2118 be based on the defined `xsd:simpleType` of the appropriate external imported code list.

2119 **Example 7-28: Supplementary Component is a Code**

```

2120 <xsd:complexType name="MeasureType">
2121     ...
2122     <xsd:simpleContent>
2123         <xsd:extension base="xsd:decimal">
2124             <xsd:attribute name="unitCode"
2125                 type="clm620:MeasureUnitCodeContent" use="optional">
2126                 ...
2127             </xsd:attribute>
2128             ...
2129         </xsd:extension>
2130     </xsd:simpleContent>
2131 </xsd:complexType>

```

2132 [R 138] If the representation term of the relevant supplementary component is an “Identifier” and
2133 validation is required, then the attribute representing this supplementary component **MUST**
2134 be based on the defined `xsd:simpleType` of the appropriate external imported identifier
2135 scheme.

2136 **Example 7-29: Supplementary component is an identifier**

```

2137 <xsd:complexType name="AmountType">
2138     <xsd:annotation>
2139         ...
2140     </xsd:annotation>
2141     <xsd:simpleContent>
2142         <xsd:extension base="xsd:decimal">
2143             <xsd:attribute name="currencyID"
2144                 type="clm54217:CurrencyIdentifierContentType"
2145 use="required">
2146                 ...
2147             </xsd:attribute>
2148         </xsd:extension>
2149     </xsd:simpleContent>
2150 </xsd:complexType>

```

2151 [R 139] If the representation term of the supplementary component is not “Code” or “Identifier”, then
2152 the attribute representing this supplementary component MUST be based on the appropriate
2153 built-in XSD data type.

2154 **Example 7-30: Supplementary component other than code or identifier**

```
2155 <xsd:complexType name="BinaryObjectType">  
2156 ...  
2157 <xsd:simpleContent>  
2158 <xsd:extension base="xsd:base64Binary">  
2159 <xsd:attribute name="format" type="xsd:string" use="optional">  
2160 ...  
2161 </xsd:attribute>  
2162 ...  
2163 </xsd:extension>  
2164 </xsd:simpleContent>  
2165 </xsd:complexType>
```

2166 **7.6.7 Restriction**

2167 The unqualified data types can be further restricted in the qualified data type module.

2168 **7.6.8 Annotation**

2169 [R 140] For every `udt:UnqualifiedDataType` `xsd:complexType` or `xsd:simpleType`
2170 definition a structured set of annotations MUST be present in the following pattern:

- 2171 • UniqueID (mandatory): The identifier that references an Unqualified Data Type instance in a
2172 unique and unambiguous way.
- 2173 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
2174 will always be UDT.
- 2175 • DictionaryEntryName (mandatory): The official name of the Unqualified Data Type.
- 2176 • VersionID (mandatory): An indication of the evolution over time of the Unqualified Data Type
2177 instance.
- 2178 • Definition (mandatory): The semantic meaning of the Unqualified Data Type.
- 2179 • RepresentationTermName (mandatory): The primary or secondary representation term of
2180 the associated Core Component Type.
- 2181 • PrimitiveType (mandatory): The primitive data type of the Unqualified Data Type.
- 2182 • BuiltInType (mandatory): The XSD built-in data type of the Unqualified Data Type.
- 2183 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
2184 applicable to the Unqualified Data Type.
- 2185 • Example (optional, repetitive): Example of a possible value of an Unqualified Data Type.

2186 **Example 7-31: Annotation of unqualified type definition**

```
2187 .. see complex type definition ...  
2188 <xsd:annotation>  
2189 <xsd:documentation xml:lang="en">  
2190 <ccts:UniqueID>UNDT000001</ccts:UniqueID>  
2191 <ccts:CategoryCode>UDT</ccts:CategoryCode>  
2192 <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>  
2193 <ccts:VersionID>1.0</ccts:VersionID>  
2194 <ccts:Definition> A number of monetary units specified in a  
2195 currency where the unit of the currency is explicit or  
2196 implied.</ccts:Definition>  
2197 <ccts:RepresentationTermName>Amount</ccts:RepresentationTermName>  
2198 <ccts:PrimitiveType>decimal</ccts:PrimitiveType>  
2199 <ccts:BuiltInType>decimal</ccts:BuiltIn  
2200 Type>  
2201 </xsd:documentation>  
2202 </xsd:annotation>  
2203 ... see complex type definition ...
```

2204 [R 141] For every supplementary component `xsd:attribute` declaration a structured set of
2205 annotations MUST be present in the following pattern:

- UniqueID (mandatory): The identifier that references a Supplementary Component instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be SC.
- Dictionary Entry Name (mandatory): The official name of the Supplementary Component.
- Definition (mandatory): The semantic meaning of the Supplementary Component.
- ObjectClassTermName (mandatory): The Object Class of the Supplementary Component.
- PropertyTermName (mandatory): The Property Term of the Supplementary Component.
- RepresentationTermName (mandatory): The Representation term of the Supplementary Component.
- UsageRule (optional, repetitive): A constraint that describes specific conditions that are applicable to the Supplementary Core Component.
- Example (optional, repetitive): Example of a possible value of a Basic Core Component.

2219 Example 7-32: Annotation of a supplementary component

```
2220 ... see complex type definition ...  
2221 <xsd:attribute name="currencyID" type="iso4217:CurrencyCodeContentType"  
2222 use="required">  
2223 <xsd:annotation>  
2224 <xsd:documentation xml:lang="en">  
2225 <ccts:UniqueID>UNDT000001-SC2</ccts:UniqueID>  
2226 <ccts:CategoryCode>SC</ccts:CategoryCode>  
2227 <ccts:DictionaryEntryName>Amount. Currency. Identifier  
2228 </ccts:DictionaryEntryName>  
2229 <ccts:Definition>The currency of the amount.</ccts:Definition>  
2230 </ccts:ObjectClassTermName>Amount</ccts:ObjectClassTermName>  
2231 <ccts:PropertyTermName>Currency</ccts:PropertyTermName>  
2232 <ccts:RepresentationTermName>Identifier</ccts:Representation  
2233 TermName>  
2234 <ccts:PrimitiveType>decimal</ccts:PrimitiveType>  
2235 <ccts:BuiltInType>decimal</ccts:BuiltInType>  
2236 <ccts:UsageRule>ReferenceUNECE Rec 9, using 3-letter alphabetic  
2237 codes.</ccts:UsageRule>  
2238 </xsd:documentation>  
2239 </xsd:annotation>  
2240 </xsd:attribute>  
2241 ... see complex type definition ...
```

2242 7.7 Qualified Data Type

2243 Ensuring consistency of qualified data types with the UN/CEFACT modularity and reuse goals requires
2244 creating a single schema module that defines all qualified data types. The qualified data type schema
2245 module name must follow the UN/CEFACT module naming approach. The qualified data type schema
2246 module will be used by the reusable ABIE schema module and all root schema modules.

2247 7.7.1 Use of Qualified Data Type Module

2248 The data types defined in the unqualified data type schema module are of type `xsd:complexType` or
2249 `xsd:simpleType`. These types are intended to be suitable as the `xsd:base` type for some, but not all
2250 BBIEs represented as `xsd:elements`. As business process modelling reveals the need for specialized
2251 data types, new 'qualified' types will need to be defined. These new qualified data types must be based
2252 on an unqualified data type and must represent a semantic or technical restriction of the unqualified data
2253 type. Technical restrictions must be implemented as a `xsd:restriction` or a new `xsd:simpleType` if the
2254 supplementary components of the qualified data type map directly to the properties of a built-in XSD data
2255 type.

2256 **7.7.2 Schema Construct**

2257 The qualified data type schema will be constructed in a standardized format in order to ensure
2258 consistency and ease of use. The specific format is shown below and must adhere to the format of the
2259 relevant sections as detailed in Appendix B.

2260 **Example 7-33: Structure of qualified data type schema module**

```
2261 <?xml version="1.0" encoding="utf-8"?>
2262 <!-- ===== -->
2263 <!-- ===== QDT Qualified Data Type Schema Module ===== -->
2264 <!-- ===== -->
2265 <!--
2266     Module of      Qualified Data Type
2267     Agency:       UN/CEFACT
2268     Version:      0.3 Rev. 6
2269     Last change:  25 June 2004
2270
2271
2272     Copyright (C) UN/CEFACT (2004). All Rights Reserved.
2273
2274     ... see copyright information ...
2275
2276 -->
2277 <xsd:schema targetNamespace=
2278     ... see namespace ...
2279     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2280     elementFormDefault="qualified" attributeFormDefault="unqualified">
2281 <!-- ===== Imports ===== -->
2282 <!-- ===== -->
2283     ... see imports ...
2284 <!-- ===== Type Definitions ===== -->
2285 <!-- ===== -->
2286     ... see type definitions ...
2287 </xsd:schema>
```

2288 **7.7.3 Namespace Scheme**

2289 [R 142] The UN/CEFACT:QualifiedDataType schema module namespace MUST be represented by
2290 the token "qdt".

2291 **Example 7-34: Namespace name**

```
2292 "urn:un:unece:uncefact:data:draft:UNCEFACTQualifiedDataTypeSchemaModule:0.3.6"
2293 "
```

2294 **Example 7-35: Schema element**

```
2295 <xsd:schema targetNamespace="urn:un:unece:uncefact:data:draft:
2296     UNCEFACTQualifiedDataTypeSchemaModule:0.3.6"
2297     xmlns:udt="urn:un:unece:uncefact:data:draft:
2298     UNCEFACTUnqualifiedDataTypeSchemaModule:0.3.6"
2299     xmlns:qdt="urn:un:unece:uncefact:data:draft:
2300     UNCEFACTQualifiedDataTypeSchemaModule:0.3.6"
2301     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2302     elementFormDefault="qualified" attributeFormDefault="unqualified">
```

2303 **7.7.4 Imports and Includes**

2304 Qualified data types will be derived from data types defined in the unqualified data types, code list, and
2305 identifier list schema modules.

2306 [R 143] The `qdt:QualifiedDataType` schema module MUST import the
2307 `udt:UnqualifiedDataType` schema module

2308 [Note]

2309 If needed, relevant UN/CEFACT and external code list and identifier scheme schema
2310 modules not imported by the `udt:UnqualifiedDataType` schema module may be
2311 imported.

2312 7.7.5 Type Definitions

- 2313 [R 144] Where required to change facets of an existing `udt:UnqualifiedDataType`, a new data
2314 type MUST be defined in the `qdt:QualifiedDataType` schema module.
- 2315 [R 145] A `qdt:QualifiedDataType` MUST be based on an **unqualified data type** and add some
2316 semantic and/or technical restriction to the unqualified data type
- 2317 [R 146] The name of a `qdt:QualifiedDataType` MUST be the name of its base
2318 `udt:UnqualifiedDataType` with separators and spaces removed and with its qualifier
2319 term added.
- 2320 [R 147] Every `qdt:QualifiedDataType` based on a `udt:UnqualifiedDataType`
2321 `xsd:complexType` whose supplementary components map directly to the properties of a
2322 built-in `xsd:data type`
2323 MUST be defined as a `xsd:simpleType`
2324 MUST contain one `xsd:restriction` element
2325 MUST include a `xsd:base` attribute that defines the specific built-in XSD data type
2326 required for the content component.
- 2327 [R 148] Every `qdt:QualifiedDataType` based on a `udt:UnqualifiedDataType`
2328 `xsd:complexType` whose supplementary components do not map directly to the properties
2329 of a built-in `xsd:data type`
2330 MUST be defined as a `xsd:complexType`
2331 MUST contain one `xsd:simpleContent` element
2332 MUST contain one `xsd:extension` element
2333 MUST include the `udt:UnqualifiedDataType` as its `xsd:base` attribute
- 2334 [R 149] Every `qdt:QualifiedDataType` based on a `udt:UnqualifiedDataType`
2335 `xsd:simpleType`
2336 MUST contain one `xsd:restriction` element
2337 MUST include the `udt:UnqualifiedDataType` as its `xsd:base` attribute

2338 [Note]

2339 If a non-standard variation of the standard date time built-in data types are required, for
2340 example year month, then a qualified data type of `textType` needs to be defined, with the
2341 appropriate restriction specified, e.g. as a pattern, to specify the required format.

2342 Example 7-36: Type Definitions

```
2343 <!-- ===== Type Definitions ===== -->
2344 <!-- =====
2345 <!-- ===== Qualified Data Type based on DateTime Type ===== -->
2346 <!-- =====
2347 <!-- ===== Qualified DT: Day_ Date. Type ===== -->
2348 <!-- =====
2349 <xsd:simpleType name="DayDateType">
2350   <xsd:annotation>
2351     ... see annotation ...
2352   </xsd:annotation>
2353   <xsd:restriction base="xsd:gDay"/>
2354 </xsd:simpleType>
2355 ...
2356 <!-- ===== Qualified Data Type based on Text. Type ===== -->
2357 <!-- =====
2358 <!-- ===== Qualified DT: Description_ Text. Type ===== -->
2359 <!-- =====
2360 <xsd:complexType name="DescriptionTextType">
2361   <xsd:annotation>
2362     ... see annotation ...
2363   </xsd:annotation>
2364   <xsd:simpleContent>
2365     <xsd:restriction base="udt:TextType"/>
2366   </xsd:simpleContent>
2367 </xsd:complexType>
```

```

2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
...
<!-- ===== Qualified Data Type based on Identifier. Type ===== -->
<!-- ===== Qualified DT: Uniform Resource_ Identifier. Type ===== -->
<!-- ===== Qualified DT: Uniform Resource_ Identifier. Type ===== -->
<!-- ===== Qualified DT: Uniform Resource_ Identifier. Type ===== -->
<xsd:simpleType name="UniformResouceIdentifierType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:restriction base="xsd:anyURI"/>
</xsd:simpleType>
...
<!-- ===== Qualified DT: Country_ Identifier. Type ===== -->
<!-- ===== Qualified DT: Country_ Identifier. Type ===== -->
<xsd:simpleType name="CountryIdentifierType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:restriction base="ids53166:CountryCodeContentType"/>
</xsd:simpleType>
...

```

2389 7.7.6 Attribute and Element Declarations

2390 There will be no element declarations in the qualified data type schema module. Attribute names will
 2391 appear in the qualified data type as defined in the unqualified data type schema module with further
 2392 restrictions applied as required.

2393 7.7.7 Extension and Restriction

2394 [R 150] The `qdt:QualifiedDataType xsd:complexType` definition `xsd:simpleContent`
 2395 element MUST only restrict attributes declared in its base type, or MUST only restrict facets
 2396 equivalent to allowed supplementary components.

2397 Example 7-37: Qualified Data Type Restricting an Identification Scheme

```

2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
<xsd:complexType name="PartyIdentifierType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:restriction base="udt:IdentifierType">
      <xsd:attribute name="schemeName" use="prohibited"/>
      <xsd:attribute name="schemeAgencyName" use="prohibited"/>
      <xsd:attribute name="schemeVersionID" use="prohibited"/>
      <xsd:attribute name="schemeDataURI" use="prohibited"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>

```

2411 7.7.8 Annotation

2412 [R 151] Every `qdt:QualifiedDataType` definition MUST contain a structured set of annotations in
 2413 the following sequence and pattern:

- 2414 • UniqueID (mandatory): The identifier that references a Qualified Data Type instance in a
 2415 unique and unambiguous way.
- 2416 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
 2417 will always be QDT.
- 2418 • DictionaryEntryName (mandatory): The official name of the Qualified Data Type.
- 2419 • VersionID (mandatory): An indication of the evolution over time of the Qualified Data Type
 2420 instance.
- 2421 • Definition (mandatory): The semantic meaning of the Qualified Data Type.

- 2422 • RepresentationTermName (mandatory): The Representation Term of the Qualified Data
2423 Type.
- 2424 • PrimitiveType (mandatory): The primitive data type of the Qualified Data Type.
- 2425 • BuiltInType (mandatory): The XSD built-in data type of the Qualified Data Type.
- 2426 • Data Type Qualifier Term (mandatory): A term that qualifies the Representation Term in
2427 order to differentiate it from its underlying Unqualified Data Type and other Qualified Data
2428 Type.
- 2429 • BusinessProcessContext (optional, repetitive): The business process context for this
2430 Qualified Data Type is associated.
- 2431 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this
2432 Qualified Data Type.
- 2433 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
2434 Qualified Data Type.
- 2435 • ProductContext (optional, repetitive): The product context for this Qualified Data Type.
- 2436 • IndustryContext (optional, repetitive): The industry context for this Qualified Data Type.
- 2437 • BusinessProcessRoleContext (optional, repetitive): The role context for this Qualified Data
2438 Type.
- 2439 • SupportingRoleContext (optional, repetitive): The supporting role context for this Qualified
2440 Data Type.
- 2441 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
2442 Qualified Data Type.
- 2443 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
2444 applicable to the Qualified Data Type.
- 2445 • Example (optional, repetitive): Example of a possible value of a Qualified Data Type.

2446 Example 7-38: Annotation of qualified data types

```

2447 ... see type definition ...
2448 <xsd:annotation>
2449   <xsd:documentation xml:lang="en">
2450     <ccts:UniqueID/>
2451     <ccts:CategoryCode>QDT</ccts:CategoryCode>
2452     <ccts:DictionaryEntryName>Account_ Type_ Code.  
2453 Type</ccts:DictionaryEntryName>
2454     <ccts:VersionID>1.0</ccts:VersionID>
2455     <ccts:Definition>This code represents the type of an account.  
2456 </ccts:Definition>
2457     <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
2458     <ccts:RepresentationTermQualifier>Account</ccts:  
2459 RepresentationTermQualifier>
2460     <ccts:RepresentationTermQualifier>Type</ccts:  
2461 RepresentationTermQualifier>
2462     <ccts:PrimitiveType>string</ccts:PrimitiveType>
2463     <ccts:BuiltInType>normalizedString</ccts:BuiltInType>
2464 </xsd:documentation>
2465 </xsd:annotation>
2466 ... see type definition ...

```

2467 [R 152] For every supplementary component `xsd:attribute` declaration a structured set of
2468 annotations MUST be present in the following pattern:

- 2469 • UniqueID (mandatory): The identifier that references a Supplementary Component of a Core
2470 Component Type instance in a unique and unambiguous way.
- 2471 • CategoryCode (mandatory): The category to which the object belongs. In this case the value
2472 will always be QDT.
- 2473 • Dictionary Entry Name (mandatory): The official name of a Supplementary Component.

- 2474 • VersionID (mandatory): An indication of the evolution over time of a Supplementary
2475 Component instance.
- 2476 • Definition (mandatory): The semantic meaning of a Supplementary Component.
- 2477 • Cardinality (mandatory): Indication whether the Supplementary Component Property
2478 represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Core
2479 Component Type.
- 2480 • PropertyTermName (optional): The Property Term of the associated Supplementary
2481 Component.
- 2482 • RepresentationTermName (optional): The Representation Term of the associated
2483 Supplementary Component.
- 2484 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
2485 applicable to the Supplementary Component.
- 2486 • BusinessProcessContext (optional, repetitive): The business process with which this
2487 Supplementary Component is associated.
- 2488 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this
2489 Supplementary Component.
- 2490 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
2491 Supplementary Component.
- 2492 • ProductContext (optional, repetitive): The product context for this Supplementary
2493 Component.
- 2494 • IndustryContext (optional, repetitive): The industry context for this Supplementary
2495 Component.
- 2496 • BusinessProcessRoleContext (optional, repetitive): The role context for this Qualified Data
2497 Type.
- 2498 • SupportingRoleContext (optional, repetitive): The supporting role context for this
2499 Supplementary Component.
- 2500 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
2501 Supplementary Component.
- 2502 • Example (optional, repetitive): Example of a possible value of a Supplementary Component.

2503 7.8 Code Lists

2504 Codes are an integral component of any business to business information flow as they facilitate the ability
2505 of the flow to be machine understandable. In order for the XML instance documents to be fully validated
2506 by the parsers, any codes used within the XML document need to be available as part of the schema
2507 validation process. Many international, national and sectorial agencies create and maintain code lists
2508 relevant to their area. If required to be used within an information flow, these code lists will be stored in
2509 their own schema, and are referred to as external code lists. For example, many of the existing code lists
2510 that exist in the United Nations Code List (UNCL) will be stored as external code list schema for use
2511 within other UN/CEFACT XSD Schema.

2512 [R 153] Each UN/CEFACT maintained code list MUST be defined in its own schema module.

2513 External code lists must be used when they exist in schema module form and when they can be directly
2514 imported into a schema module.

2515 UN/CEFACT may design and use an internal code list schema where an existing external code list
2516 schema needs to be extended, or where no suitable external code list schema exists. If a code list
2517 schema is created, it should be globally scoped and designed for reuse and sharing.

2518 [R 154] Internal code list schema MUST NOT duplicate existing external code list schema when the
2519 existing ones are available to be imported.

2520 7.8.1 Schema Construct

2521 The code list schema module will follow the general pattern for all UN/CEFACT XSD schema modules.
2522 Following the generic module information, the body of the schema will consist of code list definitions of
2523 the following general form:

2524 Example 7-39: Structure of code lists

```
2525 <?xml version="1.0" encoding="UTF-8"?>
2526 <!-- ===== -->
2527 <!-- ===== Code List: Account Type Code ; UNECE ===== -->
2528 <!-- ===== -->
2529 <!--
2530     Codelist of   Account Type Code
2531     Agency:      UNECE
2532     Version:     D.01C
2533     Last change: 25 June 2004
2534
2535     Copyright (C) UN/CEFACT (2004). All Rights Reserved.
2536
2537     ... see copyright information ...
2538
2539 -->
2540 <xsd:schema targetNamespace=" ... see namespace ...
2541     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2542     elementFormDefault="qualified" attributeFormDefault="unqualified">
2543 <!-- ===== Root Element ===== -->
2544 <!-- ===== -->
2545     ... see root element declaration ...
2546 <!-- ===== Type Definitions ===== -->
2547 <!-- ===== -->
2548 <!-- ===== Code List Type Definition: Account Type Code ===== -->
2549 <!-- ===== -->
2550     ... see type definition ...
2551 </xsd:schema>
```

2552 7.8.2 Namespace Name for Code Lists

2553 The namespace name for code list is somewhat unique in order to convey some of the supplementary
2554 component information rather than including them as attributes. Specifically, the UN/CEFACT
2555 namespace structure for a namespace name of a code list should be:

```
2556 urn:un:unece:uncefact:codelist:<status>:<Code List Agency Identifier|Code
2557 List Agency Name Text>:<Code List Identification Identifier|Code List Name
2558 Text>:<Code List Version Identifier>
```

2559 Where:

- 2560 • Namespace Identifier (NID) = un
- 2561 • Namespace Specific String =
- 2562 • unece:uncefact:codelist:<status> with unece and uncefact as fixed value second and third level
2563 domains within the NID of un and the codelist as a fixed schema type.
- 2564 • Supplementary Component String for unique identifying of code lists =
2565 <Code List. Agency Identifier|Code List. Agency Name. Text>:<Code List. Identification.
2566 Identifier|Code List. Name. Text>:<Code List. Version. Identifier>

2567 [R 155] The names for namespaces MUST have the following structure while the schema is at draft
2568 status:

```
2569 urn:un:unece:uncefact:codelist:draft:<Code List Agency
2570 Identifier|Code List Agency Name Text>:<Code List Identification.
2571 Identifier|Code List Name Text>:<Code List Version. Identifier>
```

2572 Where:

2573 codelist = this token identifying the schema as a code list
2574 Code List Agency Identifier = identifies the agency that manages a code list. The default
2575
2576

2577 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.
2578 Code List Agency Name Text = the name of the agency that maintains the code list.
2579 Code List Identification Identifier = identifies a list of the respective corresponding codes.
2580 listID is
2581 only unique within the agency that manages this code list.
2582 Code List Name Text = the name of a list of codes.
2583 Code List Version Identifier = identifies the version of a code list.

2584 **Example 7-40: Namespace name of a code list with an agency and a code list identifier at draft status**

```
2585 "urn:un:unece:uncefact:codelist:draft:6:3403:D.04A"  
2586  
2587  
2588 where  
2589 6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2590 the Code List. Agency. Identifier  
2591 3403 = UN/CEFACT data element tag for Name type code representing  
2592 the Code List. Identification. Identifier  
2593 D.04A = the version of the UN/CEFACT directory
```

2594 **Example 7-41: Namespace name of proprietary code list at draft status**

```
2595 "urn:un:unece:uncefact:codelist:draft:Security_Initiative:Document_Security:1  
2596 .2"  
2597  
2598 where  
2599 SecurityInitiative = the code list agency name of a repsonsible agency, which  
2600 is not defined in UN/CEFACT data element 3055  
2601 representing the Code List. Agency. Identifier  
2602 DocumentSecurity = the value for Code List. Name. Text  
2603 1.2 = the value for Code List. Version. Identifier
```

2604 [R 156] The namespace names for schema holding specification status MUST be of the form:

```
2605 urn:un:unece:uncefact:codelist:standard:<Code List. Agency  
2606 Identifier|Code List Agency Name Text>:<Code List Identification.  
2607 Identifier|Code List Name Text>:<Code List Version Identifier>
```

2608 Where:
2609 codelist = this token identifying the schema as a code list
2610 Code List Agency Identifier = identifies the agency that manages a code list. The default
2611 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.
2612 Code List Agency Name Text = the name of the agency that maintains the code list.
2613 Code List Identification Identifier = identifies a list of the respective corresponding codes.
2614 listID is only unique within the agency that manages this code list.
2615 Code List Name Text = the name of a list of codes.
2616 Code List Version Identifier = identifies the version of a code list.

2619 **Example 7-42: Namespace name of a code list with an agency and a code list identifier at standard status**

```
2620 "urn:un:unece:uncefact:codelist:standard:6:3403:D.04A"  
2621  
2622  
2623 where  
2624 6 = the value for UN/ECE in UN/CEFACT data element 3055 representing  
2625 the Code List. Agency. Identifier  
2626 3403 = UN/CEFACT data element tag for Name status code representing  
2627 the Code List. Identification. Identifier  
2628 D.04A = the version of the UN/CEFACT directory
```

2629 **Example 7-43: Namespace name of proprietary code list at standard status**

```
2630 "urn:un:unece:uncefact:codelist:standard:Security_Initiative:Document_Securit  
2631 y:1.2"  
2632  
2633 where  
2634 SecurityInitiative = the code list agency name of a responsible agency, which  
2635 is not defined in UN/CEFACT data element 3055  
2636 representing the Code List. Agency. Identifier  
2637 DocumentSecurity = the value for Code List. Name. Text  
2638 1.2 = the value for Code List. Version. Identifier
```

2639 Versioning for code lists published by external organisations is outside our control. As UN/CEFACT
2640 published code lists and identifier list schema the value of the <Code List. Version. Identifier> will follow
2641 the same rules as for versioning of other schema modules.

2642 7.8.3 UN/CEFACT XSD Schema Namespace Token for Code Lists

2643 A unique token will be defined for each namespace of code lists. The token representing the namespace
2644 for code lists should be constructed based on the identifier of the agency maintaining the code list and the
2645 identifier of the specific code list as issued by the maintenance agency except where there is no identifier.
2646 When there is no identifier, the name for the agency and/or code list should be used instead. This will
2647 typically be true when proprietary code lists are used. This method of token construction will provide
2648 uniqueness with a reasonably short token. When the code list is used for a qualified data type with a
2649 restricted set of valid code values, the qualified data type name is required to be used to distinguish one
2650 set of restricted values from another.

2651 The agency maintaining the code list will generally be either identified by the agency code as specified in
2652 data element 3055 in the UN/CEFACT Code List directory or the agency name if the agency does not
2653 have a code value in 3055. The identifier of the specific code list will generally be the data element tag of
2654 the corresponding list in the UN/CEFACT directory. If there is no corresponding data element, then the
2655 name of the code list will be used.

[R 157] Each UN/CEFACT maintained code list schema module MUST be represented by a unique token constructed as follows:

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

with any repeated words eliminated.

2664 Example 7-44: Code list token with an agency and a code list identifier

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

2671 Example 7-45: Code list token for a qualified data type with an agency and code list identifiers

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

2679 Example 7-46: Code list token for a proprietary code list

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

2687 Based on the constructs identified in the above examples, a namespace declaration for a code list would
2688 appear as shown in Example 7-47.

2689 Example 7-47: Target namespace declaration for a code list

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

2695 [Note]

2696
2697

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

2698 7.8.4 Schema Location

2699 Schema locations of code lists are typically defined as URL based URI schemes because of resolvability
2700 limitations of URN based URI schemes. However, UN/CEFACT XSD Schema of code lists use a URN
2701 based URI scheme for namespace declarations because persistence is considered more important than
2702 resolvability. In recognition of the need for resolvability of schema location, until such time as URNs
2703 become fully resolvable, UN/CEFACT will store schema of code lists in locations identified using a URL
2704 based URI scheme aligned with the URN based URI scheme used for the namespace declaration as
2705 follows:

2706 urn:un:unece:uncefact:codelist:<status>:<Code List. Agency Identifier|Code List. Agency Name.
2707 Text>:<Code List. Identification. Identifier|Code List. Name. Text>:<Code List. Version. Identifier>

2708 [R 158] The structure for schema location of code lists MUST be:

2709 `http://www.unece.org/uncefact/codelist/<status>/<Code List. Agency`
2710 `Identifier|Code List Agency Name Text>/<Code List Identification`
2711 `Identifier|Code List Name Text>_<Code List Version Identifier>.xsd`
2712

2713 Where:

2714 schematype = a token identifying the type of schema module: `codelist`

2715 status = the status of the schema as: `draft` | `standard`

2716 Code List Agency Identifier = identifies the agency that manages a code list. The default
2717 agencies used are those from DE 3055 but roles defined in DE 3055 cannot be used.

2718 Code List Agency Name Text = the name of the agency that maintains the code list.

2719 Code List Identification Identifier = identifies a list of the respective corresponding codes.

2720 listID is only unique within the agency that manages this code list.

2721 Code List Name Text = the name of a list of codes.

2722 Code List Version Identifier = identifies the version of a code list.
2723

2724 [R 159] Each `xsd:schemaLocation` attribute declaration of a code list MUST contain a persistent
2725 and resolvable URL.

2726 [R 160] Each `xsd:schemaLocation` attribute declaration URL of a code list MUST contain an
2727 absolute path.

2728 7.8.5 Imports and Includes

2729 UN/CEFACT Code List Schema Modules are standalone schema modules and will not import or include
2730 any other schema modules.

2731 [R 161] Code List schema modules MUST not import or include any other schema modules.

2732 7.8.6 Type Definitions

2733 [R 162] Within each code list module one, and only one, named `xsd:simpleType` MUST be defined
2734 for the content component.

2735 [R 163] The name of the `xsd:simpleType` MUST be the name of root element based on the value
2736 of the code list name text with the word "ContentType" appended.

2737 Example 7-48: Simple type definition of code lists

```
2738 <!-- ===== Type Definitions ===== -->  
2739 <!-- ===== Code List Type Definition: Account Type Code ===== -->  
2740 <!-- ===== Code List Type Definition: Account Type Code ===== -->  
2741 <!-- ===== Code List Type Definition: Account Type Code ===== -->  
2742 <xsd:simpleType name="AccountTypeCodeContentType">  
2743   <xsd:restriction base="xsd:token">  
2744     <xsd:enumeration value="2">
```

2745
2746
2747
2748

```
... see enumeration ...  
</xsd:enumeration>  
</xsd:restriction>  
</xsd:simpleType>
```

2749

[R 164] The `xsd:restriction` element base attribute value MUST be set to "xsd:token".

2750
2751

[R 165] Each code in the code list MUST be expressed as an `xsd:enumeration`, where the `xsd:value` for the enumeration is the actual code value.

2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764

Example 7-49: Enumeration facet of code lists

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

2765
2766

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

2767

[R 166] Facets other than `xsd:enumeration` MUST NOT be used in the code list schema module.

2768

7.8.7 Element Declarations

2769

Each code list schema module will contain the list of enumerations allowed for a particular element.

2770

[R 167] For each code list a single root element MUST be globally declared.

2771
2772

[R 168] The name of root element MUST be based on the code list name text following the naming rules as defined in section 5.3.

2773

[R 169] The root element MUST be of a type representing the actual list of code values.

2774
2775
2776
2777
2778

Example 7-50: Root element declaration of code lists

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

2779

7.8.8 Extension and Restriction

2780
2781

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

2782

Representation of a qualified data type of code lists could be

2783

- a combination of several individual code lists using `xsd:union`

2784

- a choice between several code lists, using `xsd:choice`

2785
2786

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

2787

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

2788
2789
2790
2791
2792
2793
2794

Example 7-51: Usage of only one Code List

```
<xsd:simpleType name="TemperatureMeasureUnitCodeType">  
  <xsd:annotation>  
    ... see annotation ...  
  </xsd:annotation>  
  <xsd:restriction base="clm66411:UnitCodeContentType">  
    <xsd:length value="3"/>
```

```

2795 <xsd:enumeration value="BTU">
2796 <xsd:annotation>
2797 <xsd:documentation source="code" xml:lang="en">
2798 <ccts:CodeName>British thermal unit</ccts:CodeName>
2799 </xsd:documentation>
2800 </xsd:annotation>
2801 </xsd:enumeration>
2802 <xsd:enumeration value="CEL">
2803 <xsd:annotation>
2804 <xsd:documentation source="code" xml:lang="en">
2805 <ccts:CodeName>degree Celsius</ccts:CodeName>
2806 </xsd:documentation>
2807 </xsd:annotation>
2808 </xsd:enumeration>
2809 <xsd:enumeration value="FAH">
2810 <xsd:annotation>
2811 <xsd:documentation source="code" xml:lang="en">
2812 <ccts:CodeName>degree Fahrenheit</ccts:CodeName>
2813 </xsd:documentation>
2814 </xsd:annotation>
2815 </xsd:enumeration>
2816 </xsd:restriction>
2817 </xsd:simpleType>

```

Example 7-52: Usage of alternative Code Lists

```

2818 <xsd:complexType name="PersonPropertyCodeType">
2819 <xsd:annotation>
2820 ... see annotation ...
2821 </xsd:annotation>
2822 <xsd:choice>
2823 <xsd:element ref="clm63479:MaritalCode"/>
2824 <xsd:element ref="clm63499:GenderCode"/>
2825 </xsd:choice>
2826 </xsd:complexType>
2827

```

Example 7-53: Combination of Code Lists

```

2828 <xsd:simpleType name="AccountDutyCodeType">
2829 <xsd:annotation>
2830 ... see annotation ...
2831 </xsd:annotation>
2832 <xsd:union memberType="clm64437:AccountTypeCodeContentType"
2833 <xsd:union memberType="clm65153:DutyTaxFeeTypeCodeContentType"/>
2834 </xsd:simpleType>
2835

```

2836 7.8.9 Annotation

2837 In order to facilitate a clear and unambiguous understanding of the list of allowable codes within an
2838 element, annotations will be provided for each enumeration to provide the code name and description.

2839 [R 170] Each **xsd:enumeration** MUST include an annotation documentation providing the code
2840 name and the code description.

Example 7-54: Annotation of codes

```

2841 <xsd:enumeration value="2">
2842 <xsd:annotation>
2843 <xsd:documentation xml:lang="en">
2844 <ccts:CodeName>Budgetary account</ccts:CodeName>
2845 <ccts:CodeDescription>Code identifying a budgetary account.
2846 </ccts:CodeDescription>
2847 </xsd:documentation>
2848 </xsd:annotation>
2849 </xsd:enumeration>...
2850

```

2851 7.9 Identifier List Schema

2852 When required separate schema modules will be defined for identification schemes that have a token,
2853 and optionally a description, and that have the same functionality as a code list. In this way, XML instance
2854 documents containing these identifiers can be fully validated by the parsers. Other identifier schemes
2855 should be defined as a qualified or unqualified data type as appropriate.

2856 External identifier lists must be used when they exist in schema module form and when they can be
2857 directly imported into a schema module.

2858 UN/CEFACT may design and use an internal identifier list where an existing external identifier list needs
2859 to be extended, or where no suitable external identifier list exists. If an identifier list is created, the lists
2860 should be globally scoped and designed for reuse and sharing.

2861 [R 171] Internal identifier lists schema MUST NOT duplicate existing external identifier list schema
2862 when the existing ones are available to be imported.

2863 [R 172] Each UN/CEFACT maintained identifier list MUST be defined in its own schema module.

2864 7.9.1 Schema Construct

2865 The identifier list schema module will follow the general pattern for all UN/CEFACT XSD schema
2866 modules. Following the generic module information, the body of the schema will consist of identifier list
2867 definitions of the following general form:

2868 Example 7-55: Structure of identifier lists

```
2869 <?xml version="1.0" encoding="UTF-8"?>
2870 <!-- ===== -->
2871 <!-- ===== ISO Country Identifier - Identifier List Schema Module ===== -->
2872 <!-- ===== -->
2873 <!--
2874 Identifier of Country Identifier
2875 Agency: ISO
2876 Version: 2
2877 Last change: 25 June 2004
2878
2879 Copyright (C) UN/CEFACT (2004). All Rights Reserved.
2880
2881 ... see copyright information ...
2882
2883 -->
2884 <xsd:schema targetNamespace=" ... see namespace ...
2885 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2886 elementFormDefault="qualified" attributeFormDefault="unqualified">
2887 <!-- ===== Root Element ===== -->
2888 <!-- ===== -->
2889 ... see root element declaration ...
2890 <!-- ===== Type Definitions ===== -->
2891 <!-- ===== -->
2892 <!-- ===== Identifier List Type Definition: Country Identifier
2893 ===== -->
2894 <!-- ===== -->
2895 ... see type definition ...
2896 </xsd:schema>
```

2897 7.9.2 Namespace Name for Identifier List Schema

2898 The namespace name for identifier list is somewhat unique in order to convey some of the supplementary
2899 component information rather than including them as attributes. Specifically, the UN/CEFACT
2900 namespace structure for a namespace name of an identifier list schema should be:

```
2901 urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme Agency
2902 Identifier|Identifier Scheme Agency Name Text>:< Identifier Scheme
2903 Identifier|Identifier Scheme Name Text>:< Identifier Scheme Version.
2904 Identifier>
```

2905 Where:

- 2906 • Namespace Identifier (NID) = un
- 2907 • Namespace Specific String =
- 2908 • unece:uncefact:codelist:<status> with unece and uncefact as fixed value second and third level
2909 domains within the NID of un and the code list as a fixed schema type.

- 2910 • Supplementary Component String for unique identifying of identifier schemes =
2911 <Identifier Scheme Agency Identifier|Identifier Scheme Agency Name Text>:< Identifier
2912 Scheme Identifier|Identifier Scheme Name Text>:< Identifier Scheme Version Identifier>

2913 [R 173] The names for namespaces MUST have the following structure while the schema is at draft
2914 status:

2915
2916 `urn:un:unece:uncefact:identifierlist:draft:<Identifier Scheme.
2917 Agency Identifier|Identifier Scheme Agency Name Text>:<Identifier
2918 Scheme Identifier|Identifier Scheme Name Text>:<Identifier Scheme
2919 Version Identifier>`

2920
2921 Where:

2922 identifierlist = this token identifying the schema as an identifier scheme
2923 Identifier Scheme Agency Identifier = the identification of the agency that maintains the
2924 identification scheme.

2925 Identifier Scheme Agency Name. Text = the name of the agency that maintains the
2926 identification list.

2927 Identifier Scheme Identifier = the identification of the identification scheme.

2928 Identifier Scheme Name. Text = the name of the identification scheme.

2929 Identifier Scheme Version. Identifier = the version of the identification scheme.

2930 **Example 7-56: Namespace name of an identifier list schema with an agency and an identifier list**
2931 **schema identifier at draft status**

2932 `"urn:un:unece:uncefact:identifierlist:draft:5:4217:2001"`
2933
2934 where
2935 5 = the value for ISO in UN/CEFACT data element 3055 representing
2936 the Code List. Agency. Identifier
2937 4217 = ISO identifier scheme identifier for currency code representing
2938 the Code List. Identification. Identifier
2939 2001 = the version of the ISO currency code list.

2940 [R 174] The namespace names for identifier list schema holding specification status MUST be of the
2941 form:

2942
2943 `urn:un:unece:uncefact:identifierlist:standard:<Identifier Scheme.
2944 Agency Identifier|Identifier Scheme Agency Name Text>:<Identifier
2945 Scheme Identifier|Identifier Scheme Name Text>:<Identifier Scheme.
2946 Version Identifier>`

2947
2948 Where:

2949 identifierlist = this token identifying the schema as an identifier scheme
2950 Identifier Scheme Agency Identifier = the identification of the agency that maintains the
2951 identification scheme.

2952 Identifier Scheme Agency Name. Text = the name of the agency that maintains the
2953 identification scheme.

2954 Identifier Scheme Identifier = the identification of the identification scheme.

2955 Identifier Scheme Name. Text = the name of the identification scheme.

2956 Identifier Scheme Version. Identifier = the version of the identification scheme.

2957 **Example 7-57: Namespace of an identifier list schema with an agency and an identifier list**
2958 **schema identifier at standard status**

2959 `"urn:un:unece:uncefact:identifierlist:standard:5:4217:2001"`
2960
2961 where
2962 5 = the value for ISO in UN/CEFACT data element 3055 representing
2963 the Code List. Agency. Identifier
2964 4217 = ISO identifier scheme identifier for currency code representing
2965 the Code List. Identification. Identifier
2966 2001 = the version of the ISO currency code list.

2967 Versioning for identifier list schemas published by external organisations is outside our control. As
2968 UN/CEFACT published identifier list schema the value of the <Identifier Scheme. Version. Identifier> will
2969 follow the same rules as for versioning of other schema modules.

2970 **7.9.3 UN/CEFACT XSD Schema Namespace Token for Identifier List** 2971 **Schema**

2972 A unique token will be defined for each namespace of an identifier list schema. The token representing
2973 the namespace for identifier lists should be constructed based on the identifier of the agency maintaining
2974 the identification list and the identifier of the specific identification list as issued by the maintenance
2975 agency. This method of token construction will provide uniqueness with a reasonably short token. When
2976 the identifier list is used for a qualified data type with a restricted set of valid identifier values, the qualified
2977 data type name is required to be used to distinguish one set of restricted values from another.

2978 The agency maintaining the identification list will be either identified by the agency code as specified in
2979 data element 3055 in the UN/CEFACT directory. The identifier of the identification list will be the identifier
2980 as allocated by the identification scheme agency.

2981 [R 175] Each UN/CEFACT maintained identifier list schema module MUST be represented by a
2982 unique token constructed as follows:

```
2983 ids[Qualified data type name]<Identification Scheme Agency  
2984 Identifier><Identification Scheme Identifier>  
2985
```

2986 **Example 7-58: Identifier list token**

```
2987 Token for the ISO Country Codes would be: ids53166-1  
2988 where:  
2989 5 = the Identification Scheme Agency Identifier for ISO in codelist 3055  
2990 3166-1 = the Identification Scheme Identifier as allocated by ISO.
```

2991 Based on the constructs identified in Example 4-37, a namespace declaration for an identifier list would
2992 appear as shown in Example 4-38.

2993 **Example 7-59: Target Namespace declaration for an Identifier list**

```
2994 <xsd:schema  
2995 targetNamespace="urn:un:unece:uncefact:identifierlist:draft:5:3166-1:1997"  
2996 xmlns:ids53166-1="urn:un:unece:uncefact:identifierlist:draft:5:3166-1:1977"  
2997 xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
2998 elementFormDefault="qualified" attributeFormDefault="unqualified">
```

2999 [Note]

3000 External developers are encouraged to follow the above construct rule when customizing
3001 schema for identifier lists to ensure that there is no namespace conflict.

3002 **7.9.4 Schema Location**

3003 Schema locations of identifier list schema are typically defined as URL based URI schemes because of
3004 resolvability limitations of URN based URI schemes. However, UN/CEFACT XSD Schema of identifier
3005 lists use a URN based URI scheme for namespace declarations because persistence is considered more
3006 important than resolvability. In recognition of the need for resolvability of schema location, until such time
3007 as URNs become fully resolvable, UN/CEFACT will store schema of identifier list in locations identified
3008 using a URL based URI scheme aligned with the URN based URI scheme used for the namespace
3009 declaration as follows:

```
3010 urn:un:unece:uncefact:identifierlist:<status>:<Identifier Scheme Agency  
3011 Identifier|Identifier Scheme Agency Name Text>:< Identifier Scheme  
3012 Identifier|Identifier Scheme Name Text>:< Identifier Scheme Version.  
3013 Identifier>
```

3014 [R 176] The structure for schema location of identifier lists MUST be:

```
3015 http://www.unece.org/uncefact/identifierlist/<status>/<Identifier  
3016 Scheme Agency Identifier|Identifier Scheme Agency Name Text>/<  
3017 Identifier Scheme Identifier|Identifier Scheme Name Text>_<  
3018
```

3019 **Identifier Scheme Version Identifier>.xsd**
3020
3021 Where:
3022 schematype = a token identifying the type of schema module: identifierlist
3023 status = the status of the schema as: **draft** | **standard**
3024 Identifier Scheme. Agency Identifier = the identification of the agency that maintains the
3025 identification scheme.
3026 Identifier Scheme. Agency Name. Text = the name of the agency that maintains the
3027 identification scheme.
3028 Identifier Scheme. Identifier = the identification of the identification scheme.
3029 Identifier Scheme. Name. Text = the name of the identification scheme.
3030 Identifier Scheme. Version. Identifier = the version of the identification scheme.

3031 [R 177] Each **xsd:schemaLocation** attribute declaration of an identifier list schema MUST contain
3032 a persistent and resolvable URL.

3033 [R 178] Each **xsd:schemaLocation** attribute declaration URL of an identifier list schema MUST
3034 contain an absolute path.

3035 7.9.5 Imports and Includes

3036 UN/CEFACT Identifier List Schema Modules are standalone schema modules and will not import or
3037 include any other schema modules.

3038 [R 179] Identifier list schema modules MUST NOT import or include any other schema modules.

3039 7.9.6 Type Definitions

3040 A restriction has to be declared in order to define the content component (the simple type) as a restriction
3041 of the unqualified data type in order to comply with parser requirements. The restriction itself is the list of
3042 enumerations.

3043 [R 180] Within each identifier list schema module one, and only one, named **xsd:simpleType**
3044 MUST be defined for the content component.

3045 [R 181] The name of the **xsd:simpleType** MUST be the name of root element with the word
3046 "ContentType" appended.

3047 Example 7-60: Simple type definition of an identifier list

```
3048 <!-- ===== Type Definitions ===== -->  
3049 <!-- ===== -->  
3050 <xsd:simpleType name="CountryIdentifierContentType">  
3051   <xsd:restriction base="xsd:token">  
3052     <xsd:enumeration value="AU">  
3053       ... see enumeration ...  
3054     </xsd:enumeration>  
3055   </xsd:restriction>  
3056 </xsd:simpleType>
```

3057 [R 182] The **xsd:restriction** element base attribute value MUST be set to "xsd:token".

3058 [R 183] Each identifier in the identifier list MUST be expressed as an **xsd:enumeration**, where the
3059 **xsd:value** for the enumeration is the actual identifier value.

3060 Example 7-61: Enumeration facet of an identifier list

```
3061 ... see type definition ...  
3062 <xsd:enumeration value="AU">  
3063   <xsd:annotation>  
3064     ... see annotation  
3065   </xsd:annotation>  
3066 </xsd:enumeration>  
3067 <xsd:enumeration value="US">  
3068   <xsd:annotation>  
3069     ... see annotation  
3070 </xsd:annotation>
```

3071 `</xsd:enumeration>`
3072 `...`

3073 The purpose of the identifier list schema module is to define the list of allowable values (enumerations)
3074 that can appear within a particular element. Therefore, no other facet restrictions are allowed.

3075 [R 184] Facets other than `xsd:enumeration` MUST NOT be used in the identifier list schema
3076 module.

3077 7.9.7 Attribute and Element Declarations

3078 Each identifier list schema module will contain a list of enumerations allowed for a particular element.

3079 [R 185] For each identifier list a single root element MUST be globally declared.

3080 [R 186] The name of the root element MUST be based on the `identification scheme.name.`
3081 `text` following the naming rules as defined in section 5.3.

3082 [R 187] The root element MUST be of a type representing the actual list of identifier values.

3083 Example 7-62: Root element declaration of identifier lists

```
3084 <!-- ===== Root Element ===== -->  
3085 <!-- ===== -->  
3086 <xsd:element name="CountryIdentifier"  
3087 type="ids53166:CountryIdentifierContentType" />
```

3088 7.9.8 Extension and Restriction

3089 Users of the UN/CEFACT library may identify any subset they wish from a specific identifier list for their
3090 own trading community requirements by defining a qualified data type.

3091 Representation of a qualified data type of identifier lists could be

- 3092 • a combination of several individual identifier lists using `xsd:union`
- 3093 • a choice between several identifier lists, using `xsd:choice`

3094 Both of these can easily be accommodated in this syntax solution as required by the user's business
3095 requirements.

3096 XML declarations for using identifier lists in qualified data types are shown in the following examples.

3097 Example 7-63: Enumeration facet of identifier scheme

```
3098 ... see type definition ...  
3099 <xsd:enumeration value="AD">  
3100 <xsd:annotation>  
3101 ... see annotation ...  
3102 </xsd:annotation>  
3103 </xsd:enumeration>  
3104 <xsd:enumeration value="AE">  
3105 <xsd:annotation>  
3106 ... see annotation ...  
3107 </xsd:annotation>  
3108 </xsd:enumeration>  
3109 <xsd:enumeration value="AF">  
3110 <xsd:annotation>  
3111 ... see annotation ...  
3112 </xsd:annotation>  
3113 </xsd:enumeration>  
3114 ... see type definition ...
```

3115 Example 7-64: Usage of only one identifier scheme

```
3116 <xsd:simpleType name="CountryIdentifierType">  
3117 <xsd:annotation>  
3118 ... see annotation ...  
3119 </xsd:annotation>  
3120 <xsd:restriction base="ids53166:CountryIdentifierContentType" />  
3121 </xsd:simpleType>
```

3122 Example 7-65: Usage of alternative identifier schemes


```
3123 <xsd:complexType name="GeopoliticalIdentifierType">
3124 <xsd:annotation>
3125   ... see annotation ...
3126 </xsd:annotation>
3127 <xsd:choice>
3128   <xsd:element ref="ids53166:CountryCode"/>
3129   <xsd:element ref="ids53166-2:RegionCode"/>
3130 </xsd:choice>
3131 </xsd:complexType>
```

3132 7.9.9 Annotation

3133 In order to facilitate a clear and unambiguous understanding of the list of allowable identifiers within an
3134 element, annotations will be provided for each enumeration to provide the name, and optionally a
3135 description, of the identifier.

3136 [R 188] Each **xsd:enumeration** MUST include an annotation documentation providing the identifier
3137 name and optionally the description of the identifier.

3138 Example 7-66: Annotation of Identifiers

```
3139 <xsd:enumeration value="AU">
3140 <xsd:annotation>
3141   <xsd:documentation xml:lang="en">
3142     <ccd:IdentifierName>Australia</ccd:IdentifierName>
3143   </xsd:documentation>
3144 </xsd:annotation>
3145 </xsd:enumeration>
```

3146 **8 XML Instance Documents**

3147 In order to be UN/CEFACT conformant, an instance document must be valid against the relevant
3148 UN/CEFACT compliant XML schema. The XML instance documents should be readable and
3149 understandable by both humans and applications, and should enable reasonably intuitive interactions. It
3150 should represent all truncated tag names as described in section 7. A XPath navigation path should
3151 describe the complete semantic understanding by concatenating the nested elements. This navigation
3152 path should also reflect the meaning of each dictionary entry name of a BBIE or ASBIE.

3153 **8.1 Character Encoding**

3154 In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group
3155 (MOUMG) Resolution 01/08 (MOU/MG01n83) as agreed to by UN/CEFACT, all UN/CEFACT XML will be
3156 instantiated using UTF. UTF-8 is the preferred encoding, but UTF-16 may be used where necessary to
3157 support other languages.

3158 [R 189] All UN/CEFACT XML MUST be instantiated using UTF. UTF-8 should be used as the
3159 preferred encoding. If UTF-8 is not used, UTF-16 MUST be used.

3160 **8.2 Empty Content**

3161 Empty elements do not provide the level of assurance necessary for business information exchanges and
3162 as such, will not be used.

3163 [R 190] UN/CEFACT conformant instance documents MUST NOT contain an element devoid of
3164 content.

3165 [R 191] The `xsi:nil` attribute MUST NOT appear in any conforming instance.

3166 **8.3 xsi:type**

3167 The `xsi:type` attribute allows for substitution during an instantiation of a xml document. In the same
3168 way that substitution groups are not allowed, the `xsi:type` attribute is not allowed.

3169 [R 192] The `xsi:type` attribute MUST NOT be used.

3170

Appendix A. Related Documents

3171

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

3172

- *UN/CEFACT Core Components Technical Specification, Part 8 of the ebXML Framework Version 2.01*

3173

3174

- *ebXML Technical Architecture Specification v1.04*

3175

- *OASIS/ebXML Registry Information Model v2.0*

3176

- *ebXML Requirements Specification v1.06*

3177

- *Information Technology - Metadata registries: Framework for the Specification and Standardization of Data Elements, International Standardization Organization, ISO 11179-1*

3178

3179

- *Information Technology - Metadata registries: Classification of Concepts for the Identification of Domains, International Standardization Organization, ISO 11179-2*

3180

3181

3182

- *Information Technology - Metadata registries: Registry Metamodel, International Standardization Organization, ISO 11179-3*

3183

3184

- *Information Technology - Metadata registries: Rules and Guidelines for the Formulation of Data Definitions, International Standardization Organization, ISO 11179-4*

3185

3186

- *Information Technology - Metadata registries: Naming and Identification Principles for Data Elements, International Standardization Organization, ISO 11179-5*

3187

3188

- *Information Technology - Metadata registries: Framework for the Specification and Standardization of Data Elements, International Standardization Organization, ISO 11179-6*

3189

3190 Appendix B. Overall Structure

3191 The structure of an UN/CEFACT compliant XML schema must contain one or more of the following
3192 sections as relevant. Relevant sections must appear in the order given:

- 3193 • XML Declaration
- 3194 • Schema Module Identification and Copyright Information
- 3195 • Schema Start-Tag
- 3196 • Includes
- 3197 • Imports
- 3198 • Root element
- 3199 • Type Definitions

3200 B.1 XML Declaration

3201 A UTF-8 encoding is adopted throughout all UN/CEFACT XML schema.

3202 Example B-1: XML Declaration

```
3203 <?xml version="1.0" encoding="UTF-8"?>
```

3204 B.2 Schema Module Identification and Copyright Information

3205 Example B-2: Copyright Information

```
3206 <!-- ===== -->
3207 <!-- ===== Examples Schema Module; 0.3 Rev.6 ===== -->
3208 <!-- ===== -->
3209 <!--
3210 Module:           Example
3211 Agency:           UN/CEFACT
3212 Version:          0.3 Rev. 6
3213 Last change:     25 June 2004
3214
3215 Copyright (C) UN/CEFACT (2004). All Rights Reserved.
3216
3217 This document and translations of it may be copied and furnished to others, and
3218 derivative works that comment on or otherwise explain it or assist in its
3219 implementation may be prepared, copied, published and distributed, in whole or in
3220 part, without restriction of any kind, provided that the above copyright notice and
3221 this paragraph are included on all such copies and derivative works. However, this
3222 document itself may not be modified in any way, such as by removing the copyright
3223 notice or references to UN/CEFACT, except as needed for the purpose of developing
3224 UN/CEFACT specifications, in which case the procedures for copyrights defined in the
3225 UN/CEFACT Intellectual Property Rights document must be followed, or as required to
3226 translate it into languages other than English.
3227
3228 The limited permissions granted above are perpetual and will not be revoked by
3229 UN/CEFACT or its successors or assigns.
3230
3231 This document and the information contained herein is provided on an "AS IS" basis and
3232 UN/CEFACT DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
3233 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
3234 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
3235 -->
```

3236 B.3 Schema Start-Tag

3237 The Schema Start-Tag section of an UN/CEFACT compliant XML schema must contain one or more of
3238 the below declarations as relevant. Relevant declarations must appear in the order given:

- 3239 • Version
- 3240 • Namespaces
- 3241 • targetNamespace attribute

- 3242 • xmlns:xsd attribute
- 3243 • namespace declaration for reusable ABIEs actually used in the schema
- 3244 • namespace declaration for unqualified data types actually used in the schema
- 3245 • namespace declaration for qualified data types actually used in the schema
- 3246 • namespace declaration for code lists actually used in the schema
- 3247 • namespace declaration for identifier schemes actually used in the schema
- 3248 • Form Defaults
- 3249 • elementFormDefault
- 3250 • attributeFormDefault
- 3251 • Others
- 3252 • other schema attributes with schema namespace
- 3253 • other schema attributes with non-schema namespace

3254 Example B-3: XML Schema Start Tag

```

3255 <xsd:schema
3256   targetNamespace="urn:un:unece:unfact:data:draft:UNCEFACTExamplesSchemaModule:0.3.6"
3257   xmlns:rsm="urn:un:unece:unfact:data:draft:UNCEFACTExamplesSchemaModule:0.3.6"
3258   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3259   xmlns:ram="urn:un:unece:unfact:data:draft:UNCEFACTReusableAggregateBusinessInformati
3260   onEntitySchemaModule:0.3.6"
3261   xmlns:udt="urn:un:unece:unfact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:0.
3262   3.6"
3263   xmlns:qdt="urn:un:unece:unfact:data:draft:UNCEFACTQualifiedDataTypeSchemaModule:0.3.
3264   6"
3265   xmlns:ids53166="urn:un:unece:unfact:codelist:draft:5:3166-1:1997"
3266   xmlns:ids53166-2="urn:un:unece:unfact:codelist:draft:5:3166-2:1998"
3267   xmlns:clm65153="urn:un:unece:unfact:codelist:draft:6:5153:D.01C"
3268   xmlns:clm64405="urn:un:unece:unfact:codelist:draft:6:4405:D.01C"
3269   xmlns:clm69143="urn:un:unece:unfact:codelist:draft:6:9143:D.01C"
3270   xmlns:clmPerson_Characteristic_Code63289="urn:un:unece:unfact:codelist:draft:
3271   6:3289:D.01C" xmlns:clm63479="urn:un:unece:unfact:codelist:draft:6:3479:D.01C"
3272   xmlns:clm63499="urn:un:unece:unfact:codelist:draft:6:3499:D.01C"
3273   xmlns:clm1161131="urn:un:unece:unfact:codelist:draft:11:61131:4031"
3274   xmlns:clm66411="urn:un:unece:unfact:codelist:draft:6:6411:2001"
3275   xmlns:clm54217="urn:un:unece:unfact:codelist:draft:5:4217:2001"
3276   xmlns:clm5639="urn:un:unece:unfact:codelist:draft:5:639:1988"
3277   xmlns:clm64437="urn:un:unece:unfact:codelist:draft:6:4437:D.01C"
3278
3279   elementFormDefault="qualified"
3280   attributeFormDefault="unqualified">

```

3281 B.4 Includes

3282 The Include section of an UN/CEFACT compliant XML schema must contain one or more of the below
 3283 declarations as relevant. Relevant declarations must appear in the order given:

- 3284 • Inclusion of the internal ABIE schema module if used

3285 Example B-4: Includes

```

3286 <!-- ===== -->
3287 <!-- ===== Include ===== -->
3288 <!-- ===== -->
3289 <!-- ===== Inclusion of internal ABIE ===== -->
3290 <!-- ===== -->
3291 <xsd:include
3292   namespace="urn:un:unece:unfact:data:draft:UNCEFACTInternalAggregateBusinessInformat
3293   ionEntitySchemaModule:0.3.6"
3294   schemaLocation="http://www.unece.org/unfact/data/UNCEFACTInternalAggregateBusinessIn
3295   formationEntitySchemaModule_0.3.6_draft.xsd"/>

```

3296

B.5 Imports

3297

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:

3298

3299

- Import of the reusable ABIE schema module if used

3300

- Import of the unqualified data type schema module if used

3301

- Import of the qualified data type schema module if used

3302

- Import of code list schema modules actually used

3303

- Import of identifier list schema modules actually used

3304

Example B-5: Imports

3305

```

<!-- ===== Imports ===== -->
<!-- ===== Import of reusable UN/CEFACTAggregate Business Information Entity == -->
<!-- ===== -->
<xsd:import namespace="
urn:un:unece:uncefact:data:draft:UNCEFACTReusableAggregateBusinessInformationSchemaModule:0.3.6" schemaLocation=" http://www.unece.org/uncefact/data/
UNCEFACTReusableAggregateBusinessInformationSchemaModule_0.3.6_draft.xsd"/>
<!-- ===== Import of Unqualified Data Type ===== -->
<!-- ===== -->
<xsd:import
namespace="urn:un:unece:uncefact:data:draft:UNCEFACTUnqualifiedDataTypeSchemaModule:0.3.6" schemaLocation=" http://www.unece.org/uncefact/data/
UNCEFACTUnqualifiedDataTypeSchemaModule_0.3.6_draft.xsd"/>
<!-- ===== Import of Qualified Data Type ===== -->
<!-- ===== -->
<xsd:import
namespace="urn:un:unece:uncefact:data:draft:UNCEFACTQualifiedDataTypeSchemaModule:0.3.6" schemaLocation=" http://www.unece.org/uncefact/data/
UNCEFACTQualifiedDataTypeSchemaModule_0.3.6_draft.xsd"/>
<!-- ===== Import of Code lists ===== -->
<!-- ===== -->
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:4437:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/64437_D.01C_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:6411:2001"
schemaLocation=" http://www.unece.org/uncefact/codelist/66411_2001_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:5:4217:2001"
schemaLocation=" http://www.unece.org/uncefact/codelist/54217_2001_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:5:639-1:1988"
schemaLocation="http://www.unece.org/uncefact/codelist/5639-1.1988_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:11:61131:4031"
schemaLocation="http://www.unece.org/uncefact/codelist/1161131_4031_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:3499:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/63499_D.01C_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:3479:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/63479_D.01C_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:3289:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/63289_D.01C_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:9143:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/69143_D.01C_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:4405:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/64405_D.01C_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:6:5153:D.01C"
schemaLocation="http://www.unece.org/uncefact/codelist/65153_D.01C_draft.xsd"/>
<!-- ===== Import of Identifier Schemes ===== -->
<!-- ===== -->
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:5:3166-1:1997"
schemaLocation="http://www.unece.org/uncefact/identifierlist/53166-1_1997_draft.xsd"/>
<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:5:3166-2:1998"
schemaLocation="http://www.unece.org/uncefact/identifierlist/53166-2_1998_draft.xsd"/>

```

3359

3360

B.6 Root element

3361

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.

3362

3363

Example B-6:

3364

```

<!-- ===== -->
<!-- ===== Root element ===== -->
<!-- ===== -->
<xsd:element name="PurchaseOrder" type="exp:PurchaseOrderType">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:UniqueID>UNM0000001</ccts:UniqueID>
      <ccts:CategoryCode>RSM</ccts:CategoryCode>
      <ccts:Name>PurchaseOrder</ccts:Name>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:Description>A document that contains information directly relating to
        the economic event of ordering products.</ccts:Description>
      <ccts:BusinessDomain>TBG1</ccts:BusinessDomain>
      <ccts:BusinessProcessContext>Purchase Order</ccts:BusinessProcessContext>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

```

3365

3366

3367

3368

3369

3370

3371

3372

3373

3374

3375

3376

3377

3378

3379

3380

3381

B.7 Type Definitions

3382

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

3383

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

3384

Example B-7:

3385

```

<!-- ===== -->
<!-- ===== Type Definitions ===== -->
<!-- ===== -->
<!-- ===== Type Definitions: Account type ===== -->
<!-- ===== -->
<xsd:complexType name="AccountType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UN00000001</ccts:UniqueID>
      <ccts:CategoryCode>ABIE</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Account. Details</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:Definition>A business arrangement whereby debits and/or credits arising
        from transactions are recorded. This could be with a bank, i.e. a financial account,
        or a trading partner offering supplies or services 'on account', i.e. a commercial
        account</ccts:Definition>
      <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="ID" type="udt:IdentifierType" minOccurs="0"
      maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          <ccts:UniqueID>UN00000002</ccts:UniqueID>
          <ccts:CategoryCode>BBIE</ccts:CategoryCode>
          <ccts:DictionaryEntryName>Account.
            Identifier</ccts:DictionaryEntryName>
          <ccts:VersionID>1.0</ccts:VersionID>
          <ccts:Definition>The identification of a specific
            account.</ccts:Definition>
          <ccts:Cardinality>0..n</ccts:Cardinality>
          <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
          <ccts:PropertyTermName>Identifier</ccts:PropertyTermName>
          <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
          <ccts:BusinessTermName>Account Number</ccts:BusinessTermName>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="Status" type="ram:StatusType" minOccurs="0"
      maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">

```

3386

3387

3388

3389

3390

3391

3392

3393

3394

3395

3396

3397

3398

3399

3400

3401

3402

3403

3404

3405

3406

3407

3408

3409

3410

3411

3412

3413

3414

3415

3416

3417

3418

3419

3420

3421

3422

3423

3424

3425

3426

3427

```

3428         <ccts:UniqueID>UN00000003</ccts:UniqueID>
3429         <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
3430         <ccts:DictionaryEntryName>Account. Status</ccts:DictionaryEntryName>
3431         <ccts:VersionID>1.0</ccts:VersionID>
3432         <ccts:Definition>Status information related to account
3433 details.</ccts:Definition>
3434         <ccts:Cardinality>0..n</ccts:Cardinality>
3435         <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3436         <ccts:PropertyTermName>Status</ccts:PropertyTermName>
3437         <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
3438
3439     <ccts:AssociatedObjectClassTermName>Status</ccts:AssociatedObjectClassTermName>
3440     </xsd:documentation>
3441 </xsd:annotation>
3442 </xsd:element>
3443 <xsd:element name="Name" type="udt:NameType" minOccurs="0"
3444 maxOccurs="unbounded">
3445     <xsd:annotation>
3446         <xsd:documentation xml:lang="en">
3447             <ccts:UniqueID>UN00000004</ccts:UniqueID>
3448             <ccts:CategoryCode>BBIE</ccts:CategoryCode>
3449             <ccts:DictionaryEntryName>Account. Name.
3450 Text</ccts:DictionaryEntryName>
3451             <ccts:VersionID>1.0</ccts:VersionID>
3452             <ccts:Definition>The text name for a specific account</ccts:Definition>
3453             <ccts:Cardinality>0..n</ccts:Cardinality>
3454
3455             <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3456             <ccts:PropertyTermName>Name</ccts:PropertyTermName>
3457             <ccts:RepresentationTermName>Text</ccts:RepresentationTermName>
3458         </xsd:documentation>
3459     </xsd:annotation>
3460 </xsd:element>
3461 <xsd:element name="CurrencyCode" type="qdt:CurrencyCodeType" minOccurs="0"
3462 maxOccurs="unbounded">
3463     <xsd:annotation>
3464         <xsd:documentation xml:lang="en">
3465             <ccts:UniqueID>UN00000005</ccts:UniqueID>
3466             <ccts:CategoryCode>BBIE</ccts:CategoryCode>
3467             <ccts:DictionaryEntryName>Account. Currency.
3468 Code</ccts:DictionaryEntryName>
3469             <ccts:VersionID>1.0</ccts:VersionID>
3470             <ccts:Definition>A code specifying the currency in which monies are
3471 held within the account.</ccts:Definition>
3472             <ccts:Cardinality>0..n</ccts:Cardinality>
3473             <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3474             <ccts:PropertyTermName>Currency</ccts:PropertyTermName>
3475             <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
3476         </xsd:documentation>
3477     </xsd:annotation>
3478 </xsd:element>
3479 <xsd:element name="TypeCode" type="qdt:AccountTypeCodeType" minOccurs="0"
3480 maxOccurs="unbounded">
3481     <xsd:annotation>
3482         <xsd:documentation xml:lang="en">
3483             <ccts:UniqueID>UN00000006</ccts:UniqueID>
3484             <ccts:CategoryCode>BBIE</ccts:CategoryCode>
3485             <ccts:DictionaryEntryName>Account. Type.
3486 Code</ccts:DictionaryEntryName>
3487             <ccts:VersionID>1.0</ccts:VersionID>
3488             <ccts:Definition>This provides the ability to indicate what type of
3489 account this is (checking, savings, etc).</ccts:Definition>
3490             <ccts:Cardinality>0..1</ccts:Cardinality>
3491             <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3492             <ccts:PropertyTermName>Type</ccts:PropertyTermName>
3493             <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
3494         </xsd:documentation>
3495     </xsd:annotation>
3496 </xsd:element>
3497 <xsd:element name="Country" type="ram:CountryType" minOccurs="0"
3498 maxOccurs="unbounded">
3499     <xsd:annotation>
3500         <xsd:documentation xml:lang="en">
3501             <ccts:UniqueID>UN00000007</ccts:UniqueID>
3502             <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
3503             <ccts:DictionaryEntryName>Account. Country</ccts:DictionaryEntryName>

```



```

3504         <ccts:VersionID>1.0</ccts:VersionID>
3505         <ccts:Definition>Country information related to account
3506 details.</ccts:Definition>
3507         <ccts:Cardinality>0..n<ccts:Cardinality>
3508         <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3509         <ccts:PropertyTermName>Country</ccts:PropertyTermName>
3510
3511     <ccts:AssociatedObjectClassTermName>Country</ccts:AssociatedObjectClassTermName>
3512     </xsd:documentation>
3513     </xsd:annotation>
3514 </xsd:element>
3515     <xsd:element name="Person" type="ram:PersonType" minOccurs="0"
3516 maxOccurs="unbounded">
3517         <xsd:annotation>
3518             <xsd:documentation xml:lang="en">
3519                 <ccts:UniqueID>UN00000008</ccts:UniqueID>
3520                 <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
3521                 <ccts:DictionaryEntryName>Account. Person</ccts:DictionaryEntryName>
3522                 <ccts:VersionID>1.0</ccts:VersionID>
3523                 <ccts:Definition>Associated person information related to account
3524 details. This can be used to identify multiple people related to an account, for
3525 instance, the account holder.</ccts:Definition>
3526                 <ccts:Cardinality>0..n<ccts:Cardinality>
3527                 <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3528                 <ccts:PropertyTermName>Person</ccts:PropertyTermName>
3529
3530             <ccts:AssociatedObjectClassTermName>Person</ccts:AssociatedObjectClassTermName>
3531             </xsd:documentation>
3532         </xsd:annotation>
3533     </xsd:element>
3534     <xsd:element name="Organisation" type="ram:OrganisationType" minOccurs="0"
3535 maxOccurs="unbounded">
3536         <xsd:annotation>
3537             <xsd:documentation xml:lang="en">
3538                 <ccts:UniqueID>UN00000009</ccts:UniqueID>
3539                 <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
3540                 <ccts:DictionaryEntryName>Account.
3541 Organisation</ccts:DictionaryEntryName>
3542                 <ccts:VersionID>1.0</ccts:VersionID>
3543                 <ccts:Definition>The associated organisation information related to
3544 account details. This can be used to identify multiple organisations related to this
3545 account, for instance, the account holder.</ccts:Definition>
3546                 <ccts:Cardinality>0..n<ccts:Cardinality>
3547                 <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
3548                 <ccts:PropertyTermName>Organisation</ccts:PropertyTermName>
3549
3550             <ccts:AssociatedObjectClassTermName>Organisation</ccts:AssociatedObjectClassTermName>
3551             </xsd:documentation>
3552         </xsd:annotation>
3553     </xsd:element>
3554 </xsd:sequence>
3555 </xsd:complexType>

```

3556 Example B-8: Complete Structure

```

3557 <?xml version="1.0" encoding="UTF-8"?>
3558 <!-- ===== -->
3559 <!-- ===== [MODULENAME] Schema Module; [VERSION] ===== -->
3560 <!-- ===== -->
3561 <!--
3562     Module:          [MODULENAME]
3563     Agency:         UN/CEFACT
3564     Version:        [VERSION]
3565     Last change:    [DATE OF LAST CHANGE]
3566
3567     Copyright (C) UN/CEFACT (2004). All Rights Reserved.
3568
3569     This document and translations of it may be copied and furnished to others, and
3570     derivative works that comment on or otherwise explain it or assist in its
3571     implementation may be prepared, copied, published and distributed, in whole or in
3572     part, without restriction of any kind, provided that the above copyright notice and
3573     this paragraph are included on all such copies and derivative works. However, this
3574     document itself may not be modified in any way, such as by removing the copyright
3575     notice or references to UN/CEFACT, except as needed for the purpose of developing
3576     UN/CEFACT specifications, in which case the procedures for copyrights defined in the
3577     UN/CEFACT Intellectual Property Rights document must be followed, or as required
3578     to translate it into languages other than English.

```

3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619

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

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

3620 **Appendix C. ATG Approved Acronyms and Abbreviations**

3621 The following constitutes a list of ATG approved acronyms and abbreviations which must be used within
3622 tag names when these words are part of the dictionary entry name:

3623 ID – Identifier

3624 URI – Uniform Resource Identifier

3625 **Appendix D. Core Component Schema Module**

3626 The Core Component Schema Module is published as a separate file – CoreComponentType.xsd.
3627

3628 **Appendix E. Unqualified Data Type Schema Module**

3629 The Unqualified Data Type Schema Module is published as a separate file – UnqualifiedData**Type**.xsd.

Appendix F. Annotation Templates

3631 The following templates define the annotation for each of the schema modules.

```

3632 <!-- Root Schema Documentation -->
3633     <xsd:annotation>
3634         <xsd:documentation xml:lang="en">
3635             <ccts:UniqueID></ccts:UniqueID>
3636             <ccts:CategoryCode>RSM</ccts:CategoryCode>
3637             <ccts:Name></ccts:Name>
3638             <ccts:VersionID></ccts:VersionID>
3639             <ccts:Description></ccts:Description>
3640             <ccts:BusinessDomain></ccts:BusinessDomain>
3641             <ccts:BusinessProcessContext></ccts:BusinessProcessContext>
3642             <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
3643             <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
3644             <ccts:ProductContext></ccts:ProductContext>
3645             <ccts:IndustryContext></ccts:IndustryContext>
3646             <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
3647             <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
3648             <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
3649         </xsd:documentation>
3650     </xsd:annotation>
3651
3652 <!-- ABIE Documentation -->
3653     <xsd:annotation>
3654         <xsd:documentation xml:lang="en">
3655             <ccts:UniqueID></ccts:UniqueID>
3656             <ccts:CategoryCode>ABIE</ccts:CategoryCode>
3657             <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3658             <ccts:VersionID></ccts:VersionID>
3659             <ccts:Definition></ccts:Definition>
3660             <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
3661             <ccts:ObjectClassQualifierTermName></ccts:ObjectClassQualifierTermName>
3662             <ccts:BusinessProcessContext></ccts:BusinessProcessContext>
3663             <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
3664             <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
3665             <ccts:ProductContext></ccts:ProductContext>
3666             <ccts:IndustryContext></ccts:IndustryContext>
3667             <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
3668             <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
3669             <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
3670             <ccts:UsageRule></ccts:UsageRule>
3671             <ccts:BusinessTermName></ccts:BusinessTermName>
3672             <ccts:Example></ccts:Example>
3673         </xsd:documentation>
3674     </xsd:annotation>
3675
3676 <!-- BBIE Documentation -->
3677     <xsd:annotation>
3678         <xsd:documentation xml:lang="en">
3679             <ccts:UniqueID></ccts:UniqueID>
3680             <ccts:CategoryCode>ABIE</ccts:CategoryCode>
3681             <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3682             <ccts:VersionID></ccts:VersionID>
3683             <ccts:Definition></ccts:Definition>
3684             <ccts:Cardinality></ccts:Cardinality>
3685             <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
3686             <ccts:ObjectClassQualifierTermName></ccts:ObjectClassQualifierTermName>

```

```

3687     <ccts:PropertyTermName></ccts:PropertyTermName>
3688     <ccts:PropertyQualifierTermName></ccts:PropertyQualifierTermName>
3689     <ccts:RepresentationTermName></ccts:RepresentationTermName>
3690     <ccts:BusinessProcessContext></ccts:BusinessProcessContext>
3691     <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
3692     <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
3693     <ccts:ProductContext></ccts:ProductContext>
3694     <ccts:IndustryContext></ccts:IndustryContext>
3695     <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
3696     <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
3697     <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
3698     <ccts:UsageRule></ccts:UsageRule>
3699     <ccts:BusinessTermName></ccts:BusinessTermName>
3700     <ccts:Example></ccts:Example>
3701   </xsd:documentation>
3702 </xsd:annotation>
3703
3704 <!-- ASBIE Documentation -->
3705   <xsd:annotation>
3706     <xsd:documentation xml:lang="en">
3707       <ccts:UniqueID></ccts:UniqueID>
3708       <ccts:CategoryCode>ABIE</ccts:CategoryCode>
3709       <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3710       <ccts:VersionID></ccts:VersionID>
3711       <ccts:Definition></ccts:Definition>
3712       <ccts:Cardinality></ccts:Cardinality>
3713       <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
3714       <ccts:ObjectClassQualifierTermName></ccts:ObjectClassQualifierTermName>
3715       <ccts:PropertyTermName></ccts:PropertyTermName>
3716       <ccts:PropertyQualifierTermName></ccts:PropertyQualifierTermName>
3717       <ccts:AssociatedObjectClassTermName></ccts:AssociatedObjectClassTermName>
3718       <ccts:AssociatedObjectClassQualifierTermName></ccts:
3719 AssociatedObjectClassQualifierTermName>
3720       <ccts:BusinessProcessContext></ccts:BusinessProcessContext>
3721       <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
3722       <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
3723       <ccts:ProductContext></ccts:ProductContext>
3724       <ccts:IndustryContext></ccts:IndustryContext>
3725       <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
3726       <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
3727       <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
3728       <ccts:UsageRule></ccts:UsageRule>
3729       <ccts:BusinessTermName></ccts:BusinessTermName>
3730       <ccts:Example></ccts:Example>
3731     </xsd:documentation>
3732   </xsd:annotation>
3733
3734 <!-- Qualified Data Type Documentation -->
3735   <xsd:annotation>
3736     <xsd:documentation xml:lang="en">
3737       <ccts:UniqueID></ccts:UniqueID>
3738       <ccts:CategoryCode>QDT</ccts:CategoryCode>
3739       <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3740       <ccts:VersionID></ccts:VersionID>
3741       <ccts:Definition></ccts:Definition>
3742       <ccts:RepresentationTermName></ccts:RepresentationTermName>
3743       <ccts:DataTypeQualifierTermName></ccts:DataTypeQualifierTermName>
3744       <ccts:PrimitiveType></ccts:PrimitiveType>
3745       <xsd:BuiltInType></xsd:BuiltInType>
3746       <ccts:BusinessProcessContext></ccts:BusinessProcessContext>

```

```

3747     <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
3748     <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
3749     <ccts:ProductContext></ccts:ProductContext>
3750     <ccts:IndustryContext></ccts:IndustryContext>
3751     <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
3752     <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
3753     <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
3754     <ccts:UsageRule></ccts:UsageRule>
3755     <ccts:BusinessTermName></ccts:BusinessTermName>
3756     <ccts:Example></ccts:Example>
3757   </xsd:documentation>
3758 </xsd:annotation>
3759
3760 <!-- Unqualified Data Type Documentation-->
3761   <xsd:annotation>
3762     <xsd:documentation xml:lang="en">
3763       <ccts:UniqueID></ccts:UniqueID>
3764       <ccts:CategoryCode>CCT</ccts:CategoryCode>
3765       <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3766       <ccts:VersionID></ccts:VersionID>
3767       <ccts:Definition></ccts:Definition>
3768       <ccts:RepresentationTermName></ccts:RepresentationTermName>
3769       <ccts:PrimitiveType></ccts:PrimitiveType>
3770       <xsd:BuiltInType></xsd:BuiltInType>
3771       <ccts:UsageRule></ccts:UsageRule>
3772       <ccts:BusinessTermName></ccts:BusinessTermName>
3773       <ccts:Example></ccts:Example>
3774     </xsd:documentation>
3775   </xsd:annotation>
3776
3777 <!-- Unqualified Data Type Supplementary Component Documentation-->
3778   <xsd:annotation>
3779     <xsd:documentation xml:lang="en">
3780       <ccts:UniqueID></ccts:UniqueID>
3781       <ccts:CategoryCode>SC</ccts:CategoryCode>
3782       <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3783       <ccts:Definition></ccts:Definition>
3784       <ccts:ObjectClassTermName></ccts: ObjectClassTermName>
3785       <ccts:PropertyTermName></ccts:PropertyTermName>
3786       <ccts:RepresentationTermName></ccts:RepresentationTermName>
3787       <ccts:ObjectClassTermeName></ccts:ObjectClassTermeName>
3788       <ccts:PrimitiveType></ccts:PrimitiveType>
3789       <xsd:BuiltInType></xsd:BuiltInType>
3790       <ccts:UsageRule></ccts:UsageRule>
3791       <ccts:Example></ccts:Example>
3792     </xsd:documentation>
3793   </xsd:annotation>
3794
3795 <!-- Core Component Type Documentation -->
3796   <xsd:annotation>
3797     <xsd:documentation xml:lang="en">
3798       <ccts:UniqueID></ccts:UniqueID>
3799       <ccts:CategoryCode>CCT</ccts:CategoryCode>
3800       <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3801       <ccts:VersionID></ccts:VersionID>
3802       <ccts:Definition></ccts:Definition>
3803       <ccts:RepresentationTermName></ccts:RepresentationTermName>
3804       <ccts:PrimitiveType></ccts:PrimitiveType>
3805       <ccts:UsageRule></ccts:UsageRule>
3806       <ccts:BusinessTermName></ccts:BusinessTermName>

```



```

3807         <ccts:Example></ccts:Example>
3808     </xsd:documentation>
3809 </xsd:annotation>
3810
3811 <!-- Core Component Type Supplementary Component Documentation-->
3812 <xsd:annotation>
3813     <xsd:documentation xml:lang="en">
3814         <ccts:UniqueID></ccts:UniqueID>
3815         <ccts:CategoryCode>SC</ccts:CategoryCode>
3816         <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
3817         <ccts:Definition></ccts:Definition>
3818         <ccts:ObjectClassTermName></ccts: ObjectClassTermName>
3819         <ccts:PropertyTermName></ccts:PropertyTermName>
3820         <ccts:RepresentationTermName></ccts:RepresentationTermName>
3821         <ccts:ObjectClassTermeName></ccts:ObjectClassTermeName>
3822         <ccts:PrimitiveType></ccts:PrimitiveType>
3823         <ccts:UsageRule></ccts:UsageRule>
3824         <ccts:Example></ccts:Example>
3825     </xsd:documentation>
3826 </xsd:annotation>
3827
3828 <!-- Code List / Identification Schema Documentation-->
3829 <xsd:annotation>
3830     <xsd:documentation xml:lang="en">
3831         <ccts:CodeName></ccts:CodeName>
3832         <ccts:CodeDescription></ccts:CodeDescription>
3833     </xsd:documentation>
3834 </xsd:annotation>

```

3835
3836

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

3837 The following table represents the mapping between the representation terms as defined in CCTS and
3838 their equivalent data types as declared in the CCT schema module and the UDT schema module.

Representation Term	Data Type for CCT	Data Type for UDT
Amount	xsd:decimal	xsd:decimal
Binary Object	xsd:base64Binary	xsd:base64Binary
Graphic		xsd:base64Binary
Sound		xsd:base64Binary
Video		xsd:base64Binary
Code	xsd:normalizedString	xsd:normalizedString
Date Time	xsd:string	xsd:dateTime
Date		xsd:date
Time		xsd:time
Identifier	xsd:normalizedString	xsd:normalizedString
Indicator	xsd:string	xsd:boolean
Measure	xsd:decimal	xsd:decimal
Value		xsd:decimal
Percent		xsd:decimal
Rate		xsd:decimal
Numeric	xsd:string	xsd:decimal
Quantity	xsd:decimal	xsd:decimal
Text	xsd:string	xsd:string
Name		xsd:string

3839

3840

Appendix H. Naming & Design Rules List

- 3841 [R 1] Conformance shall be determined through adherence to the content of normative sections,
3842 rules and definitions.
- 3843 [R 2] All UN/CEFACT XSD Schema design rules MUST be based on the *W3C XML Schema*
3844 *Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Data Types*.
- 3845 [R 3] All UN/CEFACT XSD Schema and UN/CEFACT conformant XML instance documents
3846 MUST be based on the W3C suite of technical specifications holding recommendation
3847 status.
- 3848 [R 4] UN/CEFACT XSD Schema MUST follow the standard structure defined in Appendix B.
- 3849 [R 5] Each element or attribute XML name MUST have one and only one fully qualified XPath
3850 (FQXP).
- 3851 [R 6] Element, attribute and type names MUST be composed of words in the English language,
3852 using the primary English spellings provided in the Oxford English Dictionary.
- 3853 [R 7] Lower camel case (LCC) MUST be used for naming attributes.
- 3854 [R 8] Upper camel case (UCC) MUST be used for naming elements and types.
- 3855 [R 9] Element, attribute and type names MUST be in singular form unless the concept itself is
3856 plural.
- 3857 [R 10] Element, attribute and type names MUST be drawn from the following character set: a-z
3858 and A-Z.
- 3859 [R 11] XML element, attribute and type names constructed from dictionary entry names MUST
3860 NOT include periods, spaces, or other separators; or characters not allowed by W3C XML
3861 1.0 for XML names.
- 3862 [R 12] XML element, attribute and type names MUST NOT use acronyms, abbreviations, or other
3863 word truncations, except those included in the UN/CEFACT controlled vocabulary or listed
3864 in Appendix C.
- 3865 [R 13] Acronyms and abbreviations at the beginning of an attribute declaration MUST appear in all
3866 lower case. All other acronym and abbreviation usage in an attribute declaration must
3867 appear in upper case.
- 3868 [R 14] Acronyms MUST appear in all upper case for all element declarations and type definitions.
- 3869 [R 15] All element declarations for BBIEs and ASBIEs MUST be locally declared within the parent
3870 ABIE type.
- 3871 [R 16] A root schema MUST be created for each unique business information exchange.
- 3872 [R 17] A root schema MUST NOT replicate reusable constructs available in schema modules
3873 capable of being referenced through `xsd:include` or `xsd:import`.
- 3874 [R 18] UN/CEFACT XSD schema modules MUST either be treated as external schema modules
3875 or as internal schema modules of the root schema.
- 3876 [R 19] All UN/CEFACT internal schema modules MUST be in the same namespace as their
3877 corresponding `rsm:RootSchema`.
- 3878 [R 20] Each UN/CEFACT internal schema module MUST be named
3879 `<ParentRootSchemaModuleName><InternalSchemaModuleFunction>SchemaModule`
- 3880 [R 21] A Core Component Type schema module MUST be created
- 3881 [R 22] The `cct:CoreComponentType` schema module MUST be named "CCTS CCT Schema
3882 Module"
- 3883 [R 23] An Unqualified Data Type schema module MUST be created

- 3884 [R 24] The `udt:UnqualifiedDataType` schema module MUST be named "UN/CEFACT
3885 Unqualified Data Type Schema Module"
- 3886 [R 25] A Qualified Data Type schema module MUST be created..
- 3887 [R 26] The `qdt:QualifiedDataType` schema module MUST be named "UN/CEFACT Qualified
3888 Data Type Schema Module"
- 3889 [R 27] A Reusable Aggregate Business Information Entity schema module MUST be created
- 3890 [R 28] The `ram:ReusableAggregateBusinessInformationEntity` schema module MUST
3891 be named "UN/CEFACT Reusable Aggregate Business Information Entity Schema Module"
- 3892 [R 29] Reusable Code List schema modules MUST be created to convey code list enumerations
- 3893 [R 30] The name of each `clm:CodeList` schema module MUST be of the form: <Code List
3894 Agency Identifier|Code List Agency Name><Code List Identification
3895 Identifier|Code List Name> - Code List Schema Module Where: Code List
3896 Agency Identifier = Identifies the agency that maintains the code list Code List Agency
3897 Name = Agency that maintains the code list Code List Identification Identifier = Identifies a
3898 list of the respective corresponding codes Code List Name = The name of the code list as
3899 assigned by the agency that maintains the code list
- 3900 [R 31] An Identifier List schema module MUST be created to convey enumeration values for each
3901 identifier list that requires run time validation .
- 3902 [R 32] The name of each `ids:IdentifierList` schema module MUST be of the form:
3903 <Identifier Scheme Agency Identifier|Identifier Scheme Agency
3904 Name><Identifier Scheme Identifier|Identifier Scheme Name> -
3905 Identifier List Schema Module Where: Identifier Scheme Agency Identifier = The
3906 identification of the agency that maintains the identification scheme Identifier Scheme
3907 Agency Name = Agency that maintains the identifier list Identifier Scheme Identifier = The
3908 identification of the identification scheme Identification Scheme Name = Name as assigned
3909 by the agency that maintains the identifier list
- 3910 [R 33] Imported schema modules MUST be fully conformant with the *UN/CEFACT XML Naming
3911 and Design Rules Technical Specification* and the *Core Components Technical
3912 Specification*.
- 3913 [R 34] Every UN/CEFACT defined or imported schema module MUST have a namespace
3914 declared, using the `xsd:targetNamespace` attribute.
- 3915 [R 35] Every version of a defined or imported schema module other than internal schema modules
3916 MUST have its own unique namespace.
- 3917 [R 36] UN/CEFACT published namespace declarations or contents MUST never be changed.
- 3918 [R 37] UN/CEFACT namespaces MUST be defined as Uniform Resource Names
- 3919 [R 38] The names for namespaces MUST have the following structure while the schema is at draft
3920 status:
3921 `urn:un:unece:uncefact:<schematype>:draft:<name>:<major>.[<minor>].[`
3922 `<revision>]` Where: `schematype` = a token identifying the type of schema module:
3923 `data|process|codelist|identifierlist|documentation` `name` = the name of
3924 the module (using upper camel case) `major` = the major version number. Sequentially
3925 assigned, first release starting with the number 1. `minor` = the minor version number
3926 within a major release. Sequentially assigned, first release starting with the number 0.
3927 Not applicable for code list or identifier list schema. `revision` = sequentially assigned
3928 alphanumeric character for each revision of a minor release. Only applicable where
3929 `status` = draft and schema type does not equal code list or identifier list.
- 3930 [R 39] The namespace names for schema holding specification status MUST be of the form:
3931 `urn:un:unece:uncefact:<schematype>:standard:<name>:<major>.[<minor>`
3932 `]` Where: `schematype` = a token identifying the type of schema module:
3933 `data|process|codelist|identifierlist|documentation` `name` = the name of
3934 the module `major` = the major version number, sequentially assigned, first release starting
3935 with the number 1. `minor` = the minor version number within a major release,

- 3936 sequentially assigned, first release starting with the number 0. Not applicable for code
3937 list or identifier list schema.
- 3938 [R 40] UN/CEFACT namespaces MUST only contain UN/CEFACT developed schema modules.
- 3939 [R 41] The general structure for schema location MUST be:
3940 `http://www.unece.org/unecefact/<schematype>/<name>_<major>.<minor>.`
3941 `[<revision>]_[<status>].xsd` Where: `schematype` = a token identifying the type of
3942 schema module: `data|process|codelist|identifierlist|documentation`
3943 `name` = the name of the module (using upper camel case) `major` = the major version
3944 number, sequentially assigned, first release starting with the number 1. `minor` = the
3945 minor version number within a major release, sequentially assigned, first release
3946 starting with the number 0. `revision` = sequentially assigned alphanumeric character for
3947 each revision of a minor release. Only applicable where `status` = `draft`. `status` = the
3948 status of the schema as: `draft|standard`
- 3949 [R 42] Each `xsd:schemaLocation` attribute declaration MUST contain a persistent and
3950 resolvable URL.
- 3951 [R 43] Each `xsd:schemaLocation` attribute declaration URL MUST contain an absolute path.
- 3952 [R 44] Every schema major version MUST have the URI of:
3953 `urn:un:unece:unecefact:<schematype>:<status>:<name>:<major>.0.[<revision>]`
3954
- 3955 [R 45] Every UN/CEFACT XSD Schema and schema module major version number MUST be a
3956 sequentially assigned incremental integer greater than zero.
- 3957 [R 46] Minor versioning MUST be limited to declaring new optional XSD constructs, extending
3958 existing XSD constructs and refinements of an optional nature.
- 3959 [R 47] Every UN/CEFACT XSD Schema minor version MUST have the URI of:
3960 `urn:un:unece:unecefact:cc:schema:<name>:<major>.<non-zero`
3961 `integer>.[<revision>]`
- 3962 [R 48] For UN/CEFACT minor version changes, the name of the schema construct MUST NOT
3963 change.
- 3964 [R 49] Changes in minor versions MUST NOT break semantic compatibility with prior versions.
- 3965 [R 50] UN/CEFACT minor version schema MUST incorporate all XML constructs from the
3966 immediately preceding major or minor version schema.
- 3967 [R 51] The `xsd:elementFormDefault` attribute MUST be declared and its value set to
3968 "qualified".
- 3969 [R 52] The `xsd:attributeFormDefault` attribute MUST be declared and its value set to
3970 "unqualified".
- 3971 [R 53] The "xsd" prefix MUST be used in all cases when referring to
3972 `http://www.w3.org/2001/XMLSchema` as follows:
3973 `xmlns:xsd=http://www.w3.org/2001/XMLSchema`
- 3974 [R 54] The `xsi` prefix SHALL be used where appropriate for referencing `xsd:schemaLocation`
3975 and `xsd:noNamespaceLocation` attributes in instance documents.
- 3976 [R 55] `xsd:appInfo` MUST NOT be used.
- 3977 [R 56] `xsd:notation` MUST NOT be used.
- 3978 [R 57] `xsd:wildcard` MUST NOT be used.
- 3979 [R 58] The `xsd:any` element MUST NOT be used.
- 3980 [R 59] The `xsd:any` attribute MUST NOT be used.
- 3981 [R 60] Mixed content MUST NOT be used (excluding documentation).
- 3982 [R 61] `xsd:substitutionGroup` MUST NOT be used.

- 3983 [R 62] `xsd:ID/IDREF` MUST NOT be used.
- 3984 [R 63] `xsd:key/xsd:keyref` MUST be used for information association.
- 3985 [R 64] The absence of a construct or data MUST NOT carry meaning.
- 3986 [R 65] User declared attributes MUST only be used to convey core component type (CCT)
3987 supplementary component information.
- 3988 [R 66] An attribute of a supplementary component with variable information MUST be based on
3989 the appropriate built-in XSD data type.
- 3990 [R 67] An attribute of a supplementary component which represents codes MUST be based on the
3991 `xsd:simpleType` of the appropriate code list
- 3992 [R 68] An attribute of a supplementary component which represents identifiers MUST be based on
3993 the `xsd:simpleType` of the appropriate identifier scheme.
- 3994 [R 69] The `xsd:nillable` attribute MUST NOT be used.
- 3995 [R 70] All element declarations MUST be local except for a root element that must be declared
3996 globally.
- 3997 [R 71] Empty elements MUST NOT be used.
- 3998 [R 72] The `xsd:type` of each leaf element declaration MUST be of the data type of its source
3999 business information entity (BBIE) or complex type of its source association business
4000 information entity (ASBIE).
- 4001 [R 73] The `xsd:all` element MUST NOT be used.
- 4002 [R 74] All type definitions MUST be named.
- 4003 [R 75] Data type definitions MUST NOT duplicate the functionality of an existing data type
4004 definition..
- 4005 [R 76] `xsd:extension` MUST only be used in the `cct:CoreComponentType` schema module and
4006 the `udt:UnqualifiedDataType` schema module. When used it MUST only extend a
4007 built-in XSD datatype.
- 4008 [R 77] When `xsd:restriction` is applied to a `xsd:simpleType` or `xsd:complexType` the
4009 derived construct MUST use a different name.
- 4010 [R 78] Each UN/CEFACT defined or declared construct MUST use the `xsd:annotation`
4011 element for required CCTS documentation.
- 4012 [R 79] The root schema module MUST be represented by a unique token.
- 4013 [R 80] The `rsm:RootSchema` MUST import the following schema modules: –
4014 `ram:ReusableABIE` Schema Module – `udt:UnqualifiedDataType` Schema Module
4015 – `qdt:QualifiedDataType` Schema Module
- 4016 [R 81] A `rsm:RootSchema` in one UN/CEFACT namespace that is dependent upon type
4017 definitions or element declaration defined in another namespace MUST import the
4018 `rsm:RootSchema` from that namespace.
- 4019 [R 82] A `rsm:RootSchema` in one UN/CEFACT namespace that is dependant upon type
4020 definitions or element declarations defined in another namespace MUST NOT import
4021 Schema Modules from that namespace other than the `rsm:RootSchema`.
- 4022 [R 83] The `rsm:RootSchema` MUST include any internal schema modules that reside in the root
4023 schema namespace.
- 4024 [R 84] A single global element known as the root element MUST be globally declared in a
4025 `rsm:RootSchema`.
- 4026 [R 85] The name of the root element MUST be the name of the Message Assembly with
4027 separators and spaces removed.

- 4028 [R 86] Root schema MUST define a single `xsd:complexType` that fully describes the business
4029 information exchange.
- 4030 [R 87] The name of the top-level complex type MUST be the name of the root element with the
4031 word "Type" appended.
- 4032 [R 88] The `xsd:complexType` of the root element must be the top-level complex type.
- 4033 [R 89] For every `rsm:RootSchema` root element declaration a structured set of annotations
4034 MUST be present in the following pattern:
- 4035 • UniqueID (mandatory): The identifier that references the Message Assembly instance in
4036 a unique and unambiguous way.
 - 4037 • CategoryCode (mandatory): The category to which the object belongs. In this case the
4038 value will always be RSM.
 - 4039 • Name (mandatory): The name of the Message Assembly
 - 4040 • VersionID (mandatory): An indication of the evolution over time of a Message Assembly.
 - 4041 • Description (mandatory): A brief description of the business information exchange.
 - 4042 • BusinessDomain (mandatory, repetitive): The TBG group(s) that developed this
4043 Message Assembly.
 - 4044 • BusinessProcessContext (mandatory, repetitive): The business process with which this
4045 Message Assembly is associated.
 - 4046 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for
4047 this Message Assembly.
 - 4048 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
4049 Message Assembly.
 - 4050 • ProductContext (optional, repetitive): The product context for this Message Assembly.
 - 4051 • IndustryContext (optional, repetitive): The industry context for this Message Assembly.
 - 4052 • BusinessProcessRoleContext (optional, repetitive): The role context for this Message
4053 Assembly.
 - 4054 • SupportingRoleContext (optional, repetitive): The supporting role context for this
4055 Message Assembly.
 - 4056 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
4057 Message Assembly.
- 4058 [R 90] All UN/CEFACT internal schema modules MUST be in the same namespace as their
4059 corresponding `rsm:RootSchema`.
- 4060 [R 91] The internal schema module MUST be represented by the same token as its
4061 `rsm:RootSchema`.
- 4062 [R 92] The Reusable Aggregate Business Information Entity schema module MUST be
4063 represented by the token "ram".
- 4064 [R 93] The `ram:ReusableAggregateBusinessInformationEntity` schema MUST import
4065 the following schema modules: – `udt:UnqualifiedDataType` Schema Module –
4066 `qdt:QualifiedDataType` Schema Module
- 4067 [R 94] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named
4068 `xsd:complexType` MUST be defined.
- 4069 [R 95] The name of the ABIE `xsd:complexType` MUST be the `ccts:DictionaryEntryName`
4070 with the separators removed and with the "Details" suffix replaced with "Type".
- 4071 [R 96] Every aggregate business information entity (ABIE) `xsd:complexType` definition
4072 `xsd:content` model MUST use the `xsd:sequence` and/or `xsd:choice` elements with
4073 appropriate local element declarations to reflect each property (BBIE or ASBIE) of its class.
- 4074 [R 97] Recursion of `xsd:sequence` and/or `xsd:choice` MUST NOT occur.
- 4075 [R 98] The order and cardinality of the elements within an ABIE `xsd:complexType` MUST be
4076 according to the structure of the ABIE as defined in the model.
- 4077 [R 99] For every attribute of an object class (BBIE) identified in an ABIE, a named `xsd:element`
4078 MUST be locally declared within the `xsd:complexType` representing that ABIE.

- 4079 [R 100] Each BBIE element name declaration MUST be based on the property term and qualifiers
4080 and the representation term of the basic business information entity (BBIE). If there are
4081 successive duplicate words in the property term and representation terms of the source
4082 dictionary entry name, then the duplicate words MUST be removed.
- 4083 [R 101] If the representation term of a BBIE is 'text', it MUST be removed.
- 4084 [R 102] The BBIE element MUST be based on an appropriate data type that is defined in the
4085 UN/CEFACT `qdt:QualifiedDataType` or `udt:UnqualifiedDataType` schema
4086 modules.
- 4087 [R 103] For every association (ASBIE) identified in the UN/CEFACT syntax-neutral model, a named
4088 `xsd:element` MUST be locally declared within the `xsd:complexType` representing the
4089 ABIE.
- 4090 [R 104] Each ASBIE element name declaration MUST be based on the property term and object
4091 class of the association business information entity (ASBIE). If there are successive
4092 duplicate words in the property term and object class of the associated ABIE, then the
4093 duplicate words MUST be removed.
- 4094 [R 105] The element representing an association business information entity (ASBIE) MUST be of
4095 the complex type corresponding to its associated aggregate business information (ABIE).
- 4096 [R 106] For every ABIE `xsd:complexType` definition a structured set of annotations MUST be
4097 present in the following pattern:
- 4098 • UniqueID (mandatory): The identifier that references an ABIE instance in a unique and
4099 unambiguous way.
 - 4100 • CategoryCode (mandatory): The category to which the object belongs. In this case the
4101 value will always be ABIE.
 - 4102 • DictionaryEntryName (mandatory): The official name of an ABIE.
 - 4103 • VersionID (mandatory): An indication of the evolution over time of an ABIE instance.
 - 4104 • Definition (mandatory): The semantic meaning of an ABIE.
 - 4105 • ObjectClassTermName (mandatory): The Object Class Term of the ABIE.
 - 4106 • QualifierTermName (optional): Qualifies the Object Class Term of the ABIE.
 - 4107 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
4108 applicable to the ABIE.
 - 4109 • BusinessTermName (optional, repetitive): A synonym term under which the ABIE is
4110 commonly known and used in the business.
 - 4111 • BusinessProcessContext (optional, repetitive): The business process with which this
4112 ABIE is associated.
 - 4113 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for
4114 this ABIE.
 - 4115 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
4116 ABIE.
 - 4117 • ProductContext (optional, repetitive): The product context for this ABIE.
 - 4118 • IndustryContext (optional, repetitive): The industry context for this ABIE.
 - 4119 • BusinessProcessRoleContext (optional, repetitive): The role context for this ABIE.
 - 4120 • SupportingRoleContext (optional, repetitive): The supporting role context for this ABIE.
 - 4121 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
4122 ABIE.
 - 4123 • Example (optional, repetitive): Example of a possible value of an ABIE.
- 4124 [R 107] For every BBIE `xsd:element` declaration a structured set of annotations MUST be
4125 present in the following pattern:
- 4126 • UniqueID (mandatory): The identifier that references a BBIE instance in a unique and
4127 unambiguous way.
 - 4128 • CategoryCode (mandatory): The category to which the object belongs. In this case the
4129 value will always be BBIE.
 - 4130 • Dictionary Entry Name (mandatory): The official name of the BBIE.
 - 4131 • VersionID (mandatory): An indication of the evolution over time of a BBIE instance.
 - 4132 • Definition (mandatory): The semantic meaning of the BBIE.

- 4133 • Cardinality (mandatory): Indication whether the BIE Property represents a not-applicable,
- 4134 optional, mandatory and/or repetitive characteristic of the ABIE.
- 4135 • ObjectClassTermName (mandatory): The Object Class Term of the parent ABIE.
- 4136 • ObjectClassQualifierTermName (optional): Qualifies the Object Class Term of the parent
- 4137 ABIE.
- 4138 • PropertyTermName (mandatory): The Property Term of the BBIE.
- 4139 • PropertyQualifierTermName (optional): Qualifies the Property Term of the BBIE.
- 4140 • RepresentationTermName (mandatory): The Representation Term of the BBIE.
- 4141 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4142 applicable to the BBIE.
- 4143 • BusinessProcessContext (optional, repetitive): The business process with which this
- 4144 BBIE is associated.
- 4145 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for
- 4146 this BBIE.
- 4147 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
- 4148 BBIE.
- 4149 • ProductContext (optional, repetitive): The product context for this BBIE.
- 4150 • IndustryContext (optional, repetitive): The industry context for this BBIE.
- 4151 • BusinessProcessRoleContext (optional, repetitive): The role context for this BBIE.
- 4152 • SupportingRoleContext (optional, repetitive): The supporting role context for this BBIE.
- 4153 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
- 4154 BBIE.
- 4155 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4156 applicable to this BBIE.
- 4157 • BusinessTermName (optional, repetitive): A synonym term under which the BBIE is
- 4158 commonly known and used in the business.
- 4159 • Example (optional, repetitive): Example of a possible value of a BBIE.

4160 [R 108] For every ASBIE xsd:element declaration a structured set of annotations MUST be present
 4161 in the following pattern:

- 4162 • UniqueID (mandatory): The identifier that references an ASBIE instance in a unique and
- 4163 unambiguous way.
- 4164 • CategoryCode (mandatory): The category to which the object belongs. In this case the
- 4165 value will always be ASBIE.
- 4166 • DictionaryEntryName (mandatory): The official name of the ASBIE.
- 4167 • VersionID (mandatory): An indication of the evolution over time of the ASBIE instance.
- 4168 • Definition (mandatory): The semantic meaning of the ASBIE.
- 4169 • Cardinality (mandatory): Indication whether the ASBIE Property represents a not-
- 4170 applicable, optional, mandatory and/or repetitive characteristic of the ABIE.
- 4171 • ObjectClassTermName (mandatory): The Object Class Term of the associating ABIE.
- 4172 • ObjectClassQualifierTermName (Optional): A term that qualifies the Object Class Term of
- 4173 the associating ABIE.
- 4174 • PropertyTermName (mandatory): The Property Term of the ASBIE.
- 4175 • PropertyQualifierTermName (Optional): A term that qualifies the Property Term of the
- 4176 ASBIE.
- 4177 • AssociatedObjectClassTermName (mandatory): The Object Class Term of the
- 4178 associated ABIE.
- 4179 • AssociatedObjectClassQualifierTermName (optional): Qualifies the Object Class Term of
- 4180 the associated ABIE.
- 4181 • BusinessProcessContext (optional, repetitive): The business process with which this
- 4182 ASBIE is associated.
- 4183 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for
- 4184 this ASBIE.
- 4185 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
- 4186 ASBIE.
- 4187 • ProductContext (optional, repetitive): The product context for this ASBIE.
- 4188 • IndustryContext (optional, repetitive): The industry context for this ASBIE.
- 4189 • BusinessProcessRoleContext (optional, repetitive): The role context for this ASBIE.

- 4190 • SupportingRoleContext (optional, repetitive): The supporting role context for this ASBIE.
 - 4191 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
 - 4192 ASBIE.
 - 4193 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 - 4194 applicable to the ASBIE.
 - 4195 • BusinessTermName (optional, repetitive): A synonym term under which the ASBIE is
 - 4196 commonly known and used in the business.
 - 4197 • Example (optional, repetitive): Example of a possible value of an ASBIE.
- 4198 [R 109] The core component type (CCT) schema module MUST be represented by the token "cct".
- 4199 [R 110] The `cct:CoreCoreComponentType` schema module MUST NOT include or import any
- 4200 other schema modules.
- 4201 [R 111] Every `cct:CoreComponentType` MUST be defined as a named `xsd:complexType` in
- 4202 the `cct:CoreComponentType` schema module.
- 4203 [R 112] The name of each `xsd:complexType` based on a `cct:CoreComponentType` MUST be
- 4204 the dictionary entry name of the core component type (CCT), with the separators and
- 4205 spaces removed.
- 4206 [R 113] Each `cct:CoreComponentType` `xsd:complexType` definition MUST contain one
- 4207 `xsd:simpleContent` element.
- 4208 [R 114] The `cct:CoreComponentType` `xsd:complexType` definition `xsd:simpleContent`
- 4209 element MUST contain one `xsd:extension` element. This `xsd:extension` element must
- 4210 include an XSD based attribute that defines the specific built-in XSD data type required for
- 4211 the CCT content component.
- 4212 [R 115] Within the `cct:CoreComponentType` `xsd:extension` element a `xsd:attribute` MUST
- 4213 be declared for each supplementary component pertaining to that
- 4214 `cct:CoreComponentType`.
- 4215 [R 116] Each `cct:CoreComponentType` supplementary component `xsd:attribute` "name"
- 4216 MUST be the CCTS supplementary component dictionary entry name with the separators
- 4217 and spaces removed.
- 4218 [R 117] If the object class of the supplementary component dictionary entry name contains the
- 4219 name of the representation term of the parent CCT, the duplicated object class word or
- 4220 words MUST be removed from the supplementary component `xsd:attribute` name.
- 4221 [R 118] If the object class of the supplementary component dictionary entry name contains the term
- 4222 'identification', the term 'identification' MUST be removed from the supplementary
- 4223 component `xsd:attribute` name.
- 4224 [R 119] If the representation term of the supplementary component dictionary entry name is 'text',
- 4225 the representation term MUST be removed from the supplementary component
- 4226 `xsd:attribute` name.
- 4227 [R 120] The attribute representing as supplementary component MUST be based on the
- 4228 appropriate built-in XSD data type.
- 4229 [R 121] For every `cct:CoreComponentType` `xsd:complexType` definition a structured set of
- 4230 annotations MUST be present in the following pattern:
- 4231 • UniqueID (mandatory): The identifier that references the Core Component Type instance
 - 4232 in a unique and unambiguous way.
 - 4233 • CategoryCode (mandatory): The category to which the object belongs. In this case the
 - 4234 value will always be CCT.
 - 4235 • DictionaryEntryName (mandatory): The official name of a Core Component Type.
 - 4236 • VersionID (mandatory): An indication of the evolution over time of a Core Component
 - 4237 Type instance.
 - 4238 • Definition (mandatory): The semantic meaning of a Core Component Type.
 - 4239 • RepresentationTermName (mandatory): The primary representation term of the Core
 - 4240 Component Type.

- 4241 • PrimitiveType (mandatory): The primitive data type of the Core Component Type.
 - 4242 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 - 4243 applicable to the Core Component Type.
 - 4244 • BusinessTermName (optional, repetitive): A synonym term under which the Core
 - 4245 Component Type is commonly known and used in the business.
 - 4246 • Example (optional, repetitive): Example of a possible value of a Core Component Type.
- [R 122] For every supplementary component `xsd:attribute` declaration a structured set of annotations MUST be present in the following pattern:
- 4247 • UniqueID (mandatory): The identifier that references a Supplementary Component
 - 4248 instance in a unique and unambiguous way.
 - 4249 • CategoryCode (mandatory): The category to which the object belongs. In this case the
 - 4250 value will always be SC.
 - 4251 • DictionaryEntryName (mandatory): The official name of the Supplementary Component.
 - 4252 • Definition (mandatory): The semantic meaning of the Supplementary Component.
 - 4253 • ObjectClassTermName (mandatory): The Object Class of the Supplementary
 - 4254 Component.
 - 4255 • PropertyTermName (mandatory): The Property Term of the Supplementary Component.
 - 4256 • RepresentationTermName (mandatory): The Representation term of the Supplementary
 - 4257 Component.
 - 4258 • PrimitiveType (mandatory): The primitive data type of the Supplementary Component.
 - 4259 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 - 4260 applicable to the Supplementary Core Component.
 - 4261 • Example (optional, repetitive): Example of a possible value of a Basic Core Component.
- [R 123] The Unqualified Data Type schema module namespace MUST be represented by the token "udt".
- [R 124] The `udt:UnqualifiedDataType` schema MUST NOT import any other schema modules than the following: – `ids:IdentifierList` schema modules – `clm:CodeList` schema modules
- [R 125] A `udt:UnqualifiedDataType` MUST be defined for each approved primary and secondary representation terms identified in the CCTS Permissible Representation Terms table.
- [R 126] The name of each `udt:UnqualifiedDataType` MUST be the dictionary entry name of the primary or secondary representation term, with "Type" at the end and the separators and spaces removed.
- [R 127] For every `udt:UnqualifiedDataType` whose supplementary components map directly to the properties of a built-in `xsd:dataTtpe`, the `udt:UnqualifiedDataType` MUST be defined as a named `xsd:simpleType` in the `udt:UnqualifiedDataType` schema module.
- [R 128] Every `udt:UnqualifiedDataType` defined as a `xsd:simpleType` MUST contain one `xsd:restriction` element. This `xsd:restriction` element MUST include an `xsd:base` attribute that defines the specific built-in XSD data type required for the content component.
- [R 129] For every `udt:UnqualfiedDataType` whose supplementary components are not equivalent to the properties of a built-in XSD data type, a `udt:UnqualifiedDataType` MUST be defined as an `xsd:complexType` in the `udt:UnqualifiedDataType` schema module.
- [R 130] Every `udt:UnqualifiedDataType xsd:complexType` definition MUST contain one `xsd:simpleContent` element.
- [R 131] Every `udt:UnqualifiedDataType xsd:complexType xsd:simpleContent` element MUST contain one `xsd:extension` element. This `xsd:extension` element must include an `xsd:base` attribute that defines the specific built-in XSD datatype required for the content component.

- 4293 [R 132] Within the `udt:UnqualifiedDataType xsd:complexType xsd:extension` element
4294 an `xsd:attribute` MUST be declared for each supplementary component pertaining to
4295 the underlying CCT, unless the attribute is contained in the namespace declaration.
- 4296 [R 133] Each supplementary component `xsd:attribute` name MUST be the supplementary
4297 component name with the separators and spaces removed.
- 4298 [R 134] If the object class of the supplementary component dictionary entry name contains the
4299 name of the representation term of the parent CCT, the duplicated object class word or
4300 words MUST be removed from the supplementary component `xsd:attribute` name.
- 4301 [R 135] If the object class of the supplementary component dictionary entry name contains the term
4302 'identification', the term 'identification' MUST be removed from the supplementary
4303 component `xsd:attribute` name.
- 4304 [R 136] If the representation term of the supplementary component dictionary entry name is 'text',
4305 the representation term MUST be removed from the supplementary component
4306 `xsd:attribute` name.
- 4307 [R 137] If the representation term of the relevant supplementary component is a "Code" and
4308 validation is required, then the attribute representing this supplementary component MUST
4309 be based on the defined `xsd:simpleType` of the appropriate external imported code list.
- 4310 [R 138] If the representation term of the relevant supplementary component is an "Identifier" and
4311 validation is required, then the attribute representing this supplementary component MUST
4312 be based on the defined `xsd:simpleType` of the appropriate external imported identifier
4313 scheme.
- 4314 [R 139] If the representation term of the supplementary component is not "Code" or "Identifier", then
4315 the attribute representing this supplementary component MUST be based on the
4316 appropriate built-in XSD data type.
- 4317 [R 140] For every `udt:UnqualifiedDataType xsd:complexType` or `xsd:simpleType`
4318 definition a structured set of annotations MUST be present in the following pattern:
4319 • UniqueID (mandatory): The identifier that references an Unqualified Data Type instance
4320 in a unique and unambiguous way.
4321 • CategoryCode (mandatory): The category to which the object belongs. In this case the
4322 value will always be UDT.
4323 • DictionaryEntryName (mandatory): The official name of the Unqualified Data Type.
4324 • VersionID (mandatory): An indication of the evolution over time of the Unqualified Data
4325 Type instance.
4326 • Definition (mandatory): The semantic meaning of the Unqualified Data Type.
4327 • RepresentationTermName (mandatory): The primary or secondary representation term
4328 of the associated Core Component Type.
4329 • PrimitiveType (mandatory): The primitive data type of the Unqualified Data Type.
4330 • BuiltInType (mandatory): The XSD built-in data type of the Unqualified Data Type.
4331 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
4332 applicable to the Unqualified Data Type.
4333 • Example (optional, repetitive): Example of a possible value of an Unqualified Data Type.
- 4334 [R 141] For every supplementary component `xsd:attribute` declaration a structured set of
4335 annotations MUST be present in the following pattern:
4336 • UniqueID (mandatory): The identifier that references a Supplementary Component
4337 instance in a unique and unambiguous way.
4338 • CategoryCode (mandatory): The category to which the object belongs. In this case the
4339 value will always be SC.
4340 • Dictionary Entry Name (mandatory): The official name of the Supplementary Component.
4341 • Definition (mandatory): The semantic meaning of the Supplementary Component.
4342 • ObjectClassTermName (mandatory): The Object Class of the Supplementary
4343 Component.
4344 • PropertyTermName (mandatory): The Property Term of the Supplementary Component.
4345 • RepresentationTermName (mandatory): The Representation term of the Supplementary
4346 Component.

- 4347 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
4348 applicable to the Supplementary Core Component.
4349 • Example (optional, repetitive): Example of a possible value of a Basic Core Component.
- 4350 [R 142] The UN/CEFACT:QualifiedDataType schema module namespace MUST be represented
4351 by the token "qdt".
- 4352 [R 143] The qdt:QualifiedDataType schema module MUST import the
4353 udt:UnqualifiedDataType schema module
- 4354 [R 144] Where required to change facets of an existing udt:UnqualifiedDataType, a new data
4355 type MUST be defined in the qdt:QualifiedDataType schema module.
- 4356 [R 145] A qdt:QualifiedDataType MUST be based on an unqualified data type and add some
4357 semantic and/or technical restriction to the unqualified data type
- 4358 [R 146] The name of a qdt:QualifiedDataType MUST be the name of its base
4359 udt:UnqualifiedDataType with separators and spaces removed and with its qualifier
4360 term added.
- 4361 [R 147] Every qdt:QualifiedDataType based on a udt:UnqualifiedDataType
4362 xsd:complexType whose supplementary components map directly to the properties of a
4363 built-in xsd:data type MUST be defined as a xsd:simpleType MUST contain one
4364 xsd:restriction element MUST include a xsd:base attribute that defines the
4365 specific built-in XSD data type required for the content component.
- 4366 [R 148] Every qdt:QualifiedDataType based on a udt:UnqualifiedDataType
4367 xsd:complexType whose supplementary components do not map directly to the
4368 properties of a built-in xsd:data type MUST be defined as a xsd:complexType
4369 MUST contain one xsd:simpleContent element MUST contain one xsd:extension
4370 element MUST include the udt:UnqualifiedDataType as its xsd:base attribute
- 4371 [R 149] Every qdt:QualifiedDataType based on a udt:UnqualifiedDataType
4372 xsd:simpleType MUST contain one xsd:restriction element MUST include the
4373 udt:UnqualifiedDataType as its xsd:base attribute
- 4374 [R 150] The qdt:QualifiedDataType xsd:complexType definition xsd:simpleContent
4375 element MUST only restrict attributes declared in its base type, or MUST only restrict facets
4376 equivalent to allowed supplementary components.
- 4377 [R 151] Every qdt:QualifiedDataType definition MUST contain a structured set of annotations
4378 in the following sequence and pattern:
4379 • UniqueID (mandatory): The identifier that references a Qualified Data Type instance in a
4380 unique and unambiguous way.
4381 • CategoryCode (mandatory): The category to which the object belongs. In this case the
4382 value will always be QDT.
4383 • DictionaryEntryName (mandatory): The official name of the Qualified Data Type.
4384 • VersionID (mandatory): An indication of the evolution over time of the Qualified Data
4385 Type instance.
4386 • Definition (mandatory): The semantic meaning of the Qualified Data Type.
4387 • RepresentationTermName (mandatory): The Representation Term of the Qualified Data
4388 Type.
4389 • PrimitiveType (mandatory): The primitive data type of the Qualified Data Type.
4390 • BuiltInType (mandatory): The XSD built-in data type of the Qualified Data Type.
4391 • Data Type Qualifier Term (mandatory): A term that qualifies the Representation Term in
4392 order to differentiate it from its underlying Unqualified Data Type and other Qualified Data
4393 Types.
4394 • BusinessProcessContext (optional, repetitive): The business process context for this
4395 Qualified Data Type is associated.
4396 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for
4397 this Qualified Data Type.

- 4398 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
- 4399 Qualified Data Type.
- 4400 • ProductContext (optional, repetitive): The product context for this Qualified Data Type.
- 4401 • IndustryContext (optional, repetitive): The industry context for this Qualified Data Type.
- 4402 • BusinessProcessRoleContext (optional, repetitive): The role context for this Qualified
- 4403 Data Type.
- 4404 • SupportingRoleContext (optional, repetitive): The supporting role context for this
- 4405 Qualified Data Type.
- 4406 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
- 4407 Qualified Data Type.
- 4408 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
- 4409 applicable to the Qualified Data Type.
- 4410 • Example (optional, repetitive): Example of a possible value of a Qualified Data Type.

- 4411 [R 152] For every supplementary component `xsd:attribute` declaration a structured set of
- 4412 annotations MUST be present in the following pattern:
- 4413 • UniqueID (mandatory): The identifier that references a Supplementary Component of a
 - 4414 Core Component Type instance in a unique and unambiguous way.
 - 4415 • CategoryCode (mandatory): The category to which the object belongs. In this case the
 - 4416 value will always be QDT.
 - 4417 • Dictionary Entry Name (mandatory): The official name of a Supplementary Component.
 - 4418 • VersionID (mandatory): An indication of the evolution over time of a Supplementary
 - 4419 Component instance.
 - 4420 • Definition (mandatory): The semantic meaning of a Supplementary Component.
 - 4421 • Cardinality (mandatory): Indication whether the Supplementary Component Property
 - 4422 represents a not-applicable, optional, mandatory and/or repetitive characteristic of the
 - 4423 Core Component Type.
 - 4424 • PropertyTermName (optional): The Property Term of the associated Supplementary
 - 4425 Component.
 - 4426 • RepresentationTermName (optional): The Representation Term of the associated
 - 4427 Supplementary Component.
 - 4428 • UsageRule (optional, repetitive): A constraint that describes specific conditions that are
 - 4429 applicable to the Supplementary Component.
 - 4430 • BusinessProcessContext (optional, repetitive): The business process with which this
 - 4431 Supplementary Component is associated.
 - 4432 • GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for
 - 4433 this Supplementary Component.
 - 4434 • OfficialConstraintContext (optional, repetitive): The official constraint context for this
 - 4435 Supplementary Component.
 - 4436 • ProductContext (optional, repetitive): The product context for this Supplementary
 - 4437 Component.
 - 4438 • IndustryContext (optional, repetitive): The industry context for this Supplementary
 - 4439 Component.
 - 4440 • BusinessProcessRoleContext (optional, repetitive): The role context for this Qualified
 - 4441 Data Type.
 - 4442 • SupportingRoleContext (optional, repetitive): The supporting role context for this
 - 4443 Supplementary Component.
 - 4444 • SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this
 - 4445 Supplementary Component.
 - 4446 • Example (optional, repetitive): Example of a possible value of a Supplementary
 - 4447 Component.

4448 [R 153] Each UN/CEFACT maintained code list MUST be defined in its own schema module.

4449 [R 154] Internal code list schema MUST NOT duplicate existing external code list schema when the

4450 existing ones are available to be imported.

4451 [R 155] The names for namespaces MUST have the following structure while the schema is at draft

4452 status: `urn:un:unece:uncefact:codelist:draft:<Code List Agency`

4453 `Identifier|Code List Agency Name Text>:<Code List Identification.`

4454 Identifier|Code List Name Text>:<Code List Version. Identifier>
4455 Where: codelist = this token identifying the schema as a code list Code List Agency
4456 Identifier = identifies the agency that manages a code list. The default agencies used
4457 are those from DE 3055 but roles defined in DE 3055 cannot be used. Code List Agency
4458 Name Text = the name of the agency that maintains the code list. Code List Identification
4459 Identifier = identifies a list of the respective corresponding codes. listID is only unique
4460 within the agency that manages this code list. Code List Name Text = the name of a list of
4461 codes. Code List Version Identifier = identifies the version of a code list.

4462 [R 156] The namespace names for schema holding specification status MUST be of the form:
4463 urn:un:unece:unefact:codelist:standard:<Code List. Agency
4464 Identifier|Code List Agency Name Text>:<Code List Identification.
4465 Identifier|Code List Name Text>:<Code List Version Identifier>
4466 Where: codelist = this token identifying the schema as a code list Code List Agency
4467 Identifier = identifies the agency that manages a code list. The default agencies used
4468 are those from DE 3055 but roles defined in DE 3055 cannot be used. Code List Agency
4469 Name Text = the name of the agency that maintains the code list. Code List Identification
4470 Identifier = identifies a list of the respective corresponding codes. listID is only unique
4471 within the agency that manages this code list. Code List Name Text = the name of a list of
4472 codes. Code List Version Identifier = identifies the version of a code list.

4473 [R 157] Each UN/CEFACT maintained code list schema module MUST be represented by a unique
4474 token constructed as follows: clm[Qualified data type name]<Code List
4475 Agency Identifier|Code List Agency Name Text><Code List
4476 Identification Identifier|Code List Name Text> with any repeated words
4477 eliminated.

4478 [R 158] The structure for schema location of code lists MUST be:
4479 http://www.unece.org/unefact/codelist/<status>/<Code List. Agency
4480 Identifier|Code List Agency Name Text>/<Code List Identification
4481 Identifier|Code List Name Text>_<Code List Version Identifier>.xsd
4482 Where: schematype = a token identifying the type of schema module: codelist status =
4483 the status of the schema as: draft|standard Code List Agency Identifier = identifies the
4484 agency that manages a code list. The default agencies used are those from DE 3055
4485 but roles defined in DE 3055 cannot be used. Code List Agency Name Text = the name of
4486 the agency that maintains the code list. Code List Identification Identifier = identifies a list of
4487 the respective corresponding codes. listID is only unique within the agency that
4488 manages this code list. Code List Name Text = the name of a list of codes. Code List
4489 Version Identifier = identifies the version of a code list.

4490 [R 159] Each `xsd:schemaLocation` attribute declaration of a code list MUST contain a persistent
4491 and resolvable URL.

4492 [R 160] Each `xsd:schemaLocation` attribute declaration URL of a code list MUST contain an
4493 absolute path.

4494 [R 161] Code List schema modules MUST not import or include any other schema modules.

4495 [R 162] Within each code list module one, and only one, named `xsd:simpleType` MUST be
4496 defined for the content component.

4497 [R 163] The name of the `xsd:simpleType` MUST be the name of root element based on the
4498 value of the code list name text with the word "ContentType" appended.

4499 [R 164] The `xsd:restriction` element base attribute value MUST be set to "xsd:token".

4500 [R 165] Each code in the code list MUST be expressed as an `xsd:enumeration`, where the
4501 `xsd:value` for the enumeration is the actual code value.

4502 [R 166] Facets other than `xsd:enumeration` MUST NOT be used in the code list schema
4503 module.

4504 [R 167] For each code list a single root element MUST be globally declared.

- 4505 [R 168] The name of root element MUST be based on the `code list name text` following the
4506 naming rules as defined in section 5.3.
- 4507 [R 169] The root element MUST be of a type representing the actual list of code values.
- 4508 [R 170] Each `xsd:enumeration` MUST include an annotation documentation providing the code
4509 name and the code description.
- 4510 [R 171] Internal identifier lists schema MUST NOT duplicate existing external identifier list schema
4511 when the existing ones are available to be imported.
- 4512 [R 172] Each UN/CEFACT maintained identifier list MUST be defined in its own schema module.
- 4513 [R 173] The names for namespaces MUST have the following structure while the schema is at draft
4514 status: `urn:un:unece:uncefact:identifierlist:draft:<Identifier`
4515 `Scheme. Agency Identifier|Identifier Scheme Agency Name`
4516 `Text>:<Identifier Scheme Identifier|Identifier Scheme Name`
4517 `Text>:<Identifier Scheme Version Identifier>` Where: `identifierlist` = this
4518 token identifying the schema as an identifier scheme `Identifier Scheme Agency Identifier` =
4519 the identification of the agency that maintains the identification scheme. `Identifier`
4520 `Scheme Agency Name. Text` = the name of the agency that maintains the identification
4521 list. `Identifier Scheme Identifier` = the identification of the identification scheme. `Identifier`
4522 `Scheme Name. Text` = the name of the identification scheme. `Identifier Scheme Version.`
4523 `Identifier` = the version of the identification scheme.
- 4524 [R 174] The namespace names for identifier list schema holding specification status MUST be of
4525 the form: `urn:un:unece:uncefact:identifierlist:standard:<Identifier`
4526 `Scheme. Agency Identifier|Identifier Scheme Agency Name`
4527 `Text>:<Identifier Scheme Identifier|Identifier Scheme Name`
4528 `Text>:<Identifier Scheme. Version Identifier>` Where: `identifierlist` = this
4529 token identifying the schema as an identifier scheme `Identifier Scheme Agency Identifier` =
4530 the identification of the agency that maintains the identification scheme. `Identifier`
4531 `Scheme Agency Name. Text` = the name of the agency that maintains the identification
4532 scheme. `Identifier Scheme Identifier` = the identification of the identification scheme.
4533 `Identifier Scheme Name. Text` = the name of the identification scheme. `Identifier Scheme`
4534 `Version. Identifier` = the version of the identification scheme.
- 4535 [R 175] Each UN/CEFACT maintained identifier list schema module MUST be represented by a
4536 unique token constructed as follows: `ids[Qualified data type`
4537 `name]<Identification Scheme Agency Identifier><Identification`
4538 `Scheme Identifier>`
- 4539 [R 176] The structure for schema location of identifier lists MUST be:
4540 `http://www.unece.org/uncefact/identifierlist/<status>/<Identifier`
4541 `Scheme Agency Identifier|Identifier Scheme Agency Name Text>/<`
4542 `Identifier Scheme Identifier|Identifier Scheme Name Text>_<`
4543 `Identifier Scheme Version Identifier>.xsd` Where: `schematype` = a token
4544 identifying the type of schema module: `identifierlist status` = the status of the schema as:
4545 `draft|standard` `Identifier Scheme. Agency Identifier` = the identification of the agency
4546 that maintains the identification scheme. `Identifier Scheme. Agency Name. Text` = the
4547 name of the agency that maintains the identification scheme. `Identifier Scheme.`
4548 `Identifier` = the identification of the identification scheme. `Identifier Scheme. Name. Text` =
4549 the name of the identification scheme. `Identifier Scheme. Version. Identifier` = the version of
4550 the identification scheme.
- 4551 [R 177] Each `xsd:schemaLocation` attribute declaration of an identifier list schema MUST
4552 contain a persistent and resolvable URL.
- 4553 [R 178] Each `xsd:schemaLocation` attribute declaration URL of an identifier list schema MUST
4554 contain an absolute path.
- 4555 [R 179] Identifier list schema modules MUST NOT import or include any other schema modules.
- 4556 [R 180] Within each identifier list schema module one, and only one, named `xsd:simpleType`
4557 MUST be defined for the content component.

- 4558 [R 181] The name of the `xsd:simpleType` MUST be the name of root element with the word
4559 "ContentType" appended.
- 4560 [R 182] The `xsd:restriction` element base attribute value MUST be set to "xsd:token".
- 4561 [R 183] Each identifier in the identifier list MUST be expressed as an `xsd:enumeration`, where the
4562 `xsd:value` for the enumeration is the actual identifier value.
- 4563 [R 184] Facets other than `xsd:enumeration` MUST NOT be used in the identifier list schema
4564 module.
- 4565 [R 185] For each identifier list a single root element MUST be globally declared.
- 4566 [R 186] The name of the root element MUST be based on the `identification scheme.`
4567 `name. text` following the naming rules as defined in section 5.3.
- 4568 [R 187] The root element MUST be of a type representing the actual list of identifier values.
- 4569 [R 188] Each `xsd:enumeration` MUST include an annotation documentation providing the
4570 identifier name and optionally the description of the identifier.
- 4571 [R 189] All UN/CEFACT XML MUST be instantiated using UTF . UTF-8 should be used as the
4572 preferred encoding. If UTF-8 is not used, UTF-16 MUST be used.
- 4573 [R 190] UN/CEFACT conformant instance documents MUST NOT contain an element devoid of
4574 content.
- 4575 [R 191] The `xsi:nil` attribute MUST NOT appear in any conforming instance.
- 4576 [R 192] The `xsi:type` attribute MUST NOT be used.
- 4577

Appendix I. Glossary

- 4579 *Aggregate Business Information Entity (ABIE)* – A collection of related pieces of business information that
 4580 together convey a distinct business meaning in a specific *Business Context*. Expressed in modelling
 4581 terms, it is the representation of an *Object Class*, in a specific *Business Context*.
- 4582 *Aggregate Core Component - (ACC)* – A collection of related pieces of business information that together
 4583 convey a distinct business meaning, independent of any specific *Business Context*. Expressed in
 4584 modelling terms, it is the representation of an *Object Class*, independent of any specific *Business*
 4585 *Context*.
- 4586 *Assembly Rules - Assembly Rules* group sets of unrefined *Business Information Entities* into larger
 4587 structures. *Assembly Rules* are more fully defined and explained in the *Assembly Rules Supplemental*
 4588 *Document*.
- 4589 *Association Business Information Entity (ASBIE)* - A Business Information Entity that represents a
 4590 complex business characteristic of a specific Object Class in a specific Business Context. It has a unique
 4591 Business Semantic definition. An Association Business Information Entity represents an Association
 4592 Business Information Entity Property and is therefore associated to an Aggregate Business Information
 4593 Entity, which describes its structure. An Association Business Information Entity is derived from an
 4594 Association Core Component.
- 4595 *Association Business Information Entity Property* - A Business Information Entity Property for which the
 4596 permissible values are expressed as a complex structure, represented by an Aggregate Business
 4597 Information Entity.
- 4598 *Association Core Component (ASCC)* - A *Core Component* which constitutes a complex business
 4599 characteristic of a specific *Aggregate Core Component* that represents an *Object Class*. It has a unique
 4600 *Business Semantic* definition. An *Association Core Component* represents an *Association Core*
 4601 *Component Property* and is associated to an *Aggregate Core Component*, which describes its structure.
- 4602 *Association Core Component Property* – A *Core Component Property* for which the permissible values
 4603 are expressed as a complex structure, represented by an *Aggregate Core Component*.
- 4604 *Attribute* – A named value or relationship that exists for some or all instances of some entity and is
 4605 directly associated with that instance.
- 4606 *Basic Business Information Entity (BBIE)* – A Business Information Entity that represents a singular
 4607 business characteristic of a specific Object Class in a specific Business Context. It has a unique Business
 4608 Semantic definition. A Basic Business Information Entity represents a Basic Business Information Entity
 4609 Property and is therefore linked to a Data Type, which describes its values. A Basic Business Information
 4610 Entity is derived from a Basic Core Component.
- 4611 *Basic Business Information Entity Property* – A Business Information Entity Property for which the
 4612 permissible values are expressed by simple values, represented by a Data Type.
- 4613 *Basic Core Component (BCC)* – A *Core Component* which constitutes a singular business characteristic
 4614 of a specific *Aggregate Core Component* that represents a *Object Class*. It has a unique *Business*
 4615 *Semantic* definition. A *Basic Core Component* represents a *Basic Core Component Property* and is
 4616 therefore of a *Data Type*, which defines its set of values. *Basic Core Components* function as the
 4617 properties of *Aggregate Core Components*.
- 4618 *Basic Core Component (CC) Property* – A *Core Component Property* for which the permissible values are
 4619 expressed by simple values, represented by a *Data Type*.
- 4620 *Business Context* – The formal description of a specific business circumstance as identified by the values
 4621 of a set of *Context Categories*, allowing different business circumstances to be uniquely distinguished.
- 4622 *Business Information Entity (BIE)* – A piece of business data or a group of pieces of business data with a
 4623 unique Business Semantic definition. A Business Information Entity can be a Basic Business Information
 4624 Entity (BBIE), an Association Business Information Entity (ASBIE), or an Aggregate Business Information
 4625 Entity (ABIE).
- 4626 *Business Information Entity (BIE) Property* – A business characteristic belonging to the Object Class in its
 4627 specific Business Context that is represented by an Aggregate Business Information Entity.

- 4628 *Business Libraries* – A collection of approved process models specific to a line of business (e.g.,
4629 shipping, insurance).
- 4630 *Business Process* – The Business Process as described using the UN/CEFACT Catalogue of Common
4631 Business Processes.
- 4632 *Business Process Context* – The Business Process name(s) as described using the UN/CEFACT
4633 Catalogue of Common Business Processes as extended by the user.
- 4634 *Business Process Role Context* – The actors conducting a particular Business Process, as identified in
4635 the UN/CEFACT Catalogue of Common Business Processes.
- 4636 *Business Semantic(s)* – A precise meaning of words from a business perspective.
- 4637 *Business Term* – This is a synonym under which the *Core Component* or *Business Information Entity* is
4638 commonly known and used in the business. A *Core Component* or *Business Information Entity* may have
4639 several *Business Terms* or synonyms.
- 4640 *Cardinality* – An indication whether a characteristic is optional, mandatory and/or repetitive.
- 4641 *Catalogue of Business Information Entities* – This represents the approved set of *Business Information*
4642 *Entities* from which to choose when applying the *Core Component* discovery process
- 4643 *Catalogue of Core Components* – see Core Component Catalogue.
- 4644 *CCL* – see Core Component Library.
- 4645 *Child Core Component* – A *Core Component* used as part of a larger aggregate construct.
- 4646 *Classification Scheme* – This is an officially supported scheme to describe a given *Context Category*.
- 4647 *Constraint Language* – A formal expression of actions occurring in specific *Contexts* to assemble,
4648 structurally refine, and semantically qualify *Core Components*. The result of applying the *Constraint*
4649 *Language* to a set of *Core Components* in a specific *Context* is a set of *Business Information Entities*.
- 4650 *Content Component* – Defines the *Primitive Type* used to express the content of a *Core Component*
4651 *Type*.
- 4652 *Content Component Restrictions* – The formal definition of a format restriction that applies to the possible
4653 values of a *Content Component*.
- 4654 *Context* – Defines the circumstances in which a *Business Process* may be used. This is specified by a set
4655 of *Context Categories* known as *Business Context*.
- 4656 *Context Category* – A group of one or more related values used to express a characteristic of a business
4657 circumstance.
- 4658 *Context Rules Construct* – The overall expression of a single set of rules used to apply *Context* to *Core*
4659 *Components*.
- 4660 *Controlled Vocabulary* – A supplemental vocabulary used to uniquely define potentially ambiguous words
4661 or *Business Terms*. This ensures that every word within any of the *Core Component* names and
4662 definitions is used consistently, unambiguously and accurately.
- 4663 *Core Component (CC)* – A building block for the creation of a semantically correct and meaningful
4664 information exchange package. It contains only the information pieces necessary to describe a specific
4665 concept.
- 4666 *Core Component Catalogue* – The temporary collection of all metadata about each *Core Component*
4667 discovered during the development and initial testing of this *Core Component Technical Specification*,
4668 pending the establishment of a permanent Registry/repository.
- 4669 *Core Component Dictionary* – An extract from the *Core Component Catalogue* that provides a ready
4670 reference of the *Core Component* through its *Dictionary Entry Name*, component parts, and definition.
- 4671 *Core Component Library* – The Core Component Library is the part of the registry/repository in which
4672 Core Components shall be stored as Registry Classes. The Core Component Library will contain all the
4673 Core Component Types, Basic Core Components, Aggregate Core Components, Basic Business
4674 Information Entities and Aggregate Business Information Entities.

4675 *Core Component Property* – A business characteristic belonging to the *Object Class* represented by an
4676 *Aggregate Core Component*.

4677 *Core Component Type (CCT)* – A *Core Component*, which consists of one and only one *Content*
4678 *Component*, that carries the actual content plus one or more *Supplementary Components* giving an
4679 essential extra definition to the *Content Component*. *Core Component Types* do not have *Business*
4680 *Semantics*.

4681 *Data Type* – Defines the set of valid values that can be used for a particular *Basic Core Component*
4682 *Property* or *Basic Business Information Entity Property*. It is defined by specifying restrictions on the *Core*
4683 *Component Type* that forms the basis of the *Data Type*.

4684 *Definition* – This is the unique semantic meaning of a *Core Component*, *Business Information Entity*,
4685 *Business Context* or *Data Type*.

4686 *Dictionary Entry Name* – This is the unique official name of a *Core Component*, *Business Information*
4687 *Entity*, *Business Context* or *Data Type* in the dictionary.

4688 *Geopolitical Context* – Geographic factors that influence *Business Semantics* (e.g., the structure of an
4689 address).

4690 *Industry Classification Context* – Semantic influences related to the industry or industries of the trading
4691 partners (e.g., product identification schemes used in different industries).

4692 *Information Entity* – A reusable semantic building block for the exchange of business-related information.

4693 *Lower-Camel-Case (LCC)* – a style that capitalizes the first character of each word except the first word
4694 and compounds the name.

4695 *Naming Convention* – The set of rules that together comprise how the *Dictionary entry Name* for *Core*
4696 *Components* and *Business Information Entities* are constructed.

4697 *Object Class* – The logical data grouping (in a logical data model) to which a data element belongs
4698 (ISO11179). The *Object Class* is the part of a *Core Component's Dictionary Entry Name* that represents
4699 an activity or object in a specific *Context*.

4700 *Object Class Term* – A component of the name of a *Core Component* or *Business Information Entity*
4701 which represents the *Object Class* to which it belongs.

4702 *Official Constraints Context* – Legal and governmental influences on semantics (e.g. hazardous materials
4703 information required by law when shipping goods).

4704 *Order* – In the *Constraint Language*, the *Property* on the *ContextRules Construct* that applies a sequence
4705 to the application of a set of rules. Two Rule constructs cannot have the same value for the *Property*
4706 *Order*.

4707 *Primitive Type* – Used for the representation of a value. Possible values are String, Decimal, Integer,
4708 Boolean, Date and Binary.

4709 *Product Classification Context* – Factors influencing semantics that are the result of the goods or services
4710 being exchanged, handled, or paid for, etc. (e.g. the buying of consulting services as opposed to
4711 materials)

4712 *Property* – A peculiarity common to all members of an *Object Class*.

4713 *Property Term* – A semantically meaningful name for the characteristic of the *Object Class* that is
4714 represented by the *Core Component Property*. It shall serve as basis for the *Dictionary Entry Name* of the
4715 *Basic* and *Association Core Components* that represents this *Core Component Property*.

4716 *Qualifier Term* – A word or group of words that help define and differentiate an item (e.g. a *Business*
4717 *Information Entity* or a *Data Type*) from its associated items (e.g. from a *Core Component*, a *Core*
4718 *Compont Type*, another *Business Information Entity* or another *Data Type*).

4719 *Registry Class* – The formal definition of all the information necessary to be recorded in the Registry
4720 about a *Core Component*, a *Business Information Entity*, a *Data Type* or a *Business Context*.

4721 *Representation Term* – The type of valid values for a *Basic Core Component* or *Business Information*
4722 *Entity*.

4723 *Supplementary Component* – Gives additional meaning to the Content Component in the Core
4724 Component Type.

4725 *Supplementary Component Restrictions* – The formal definition of a format restriction that applies to the
4726 possible values of a *Supplementary Component*.

4727 *Supporting Role Context* – Semantic influences related to non-partner roles (e.g., data required by a third-
4728 party shipper in an order response going from seller to buyer.)

4729 *Syntax Binding* – The process of expressing a *Business Information Entity* in a specific syntax.

4730 *System Capabilities Context* – This *Context category* exists to capture the limitations of systems (e.g. an
4731 existing back office can only support an address in a certain form).

4732 *UMM Information Entity* – A *UMM Information Entity* realizes structured business information that is
4733 exchanged by partner roles performing activities in a business transaction. Information entities include or
4734 reference other information entities through associations.”

4735 *Unique Identifier* – The identifier that references a *Registry Class* instance in a universally unique and
4736 unambiguous way.

4737 *Upper-Camel-Case (UCC)* – a style that capitalizes the first character of each word and compounds the
4738 name.

4739 *Usage Rules* – *Usage Rules* describe how and/or when to use the *Registry Class*.

4740 *User Community* – A *User Community* is a group of practitioners, with a publicised contact address, who
4741 may define *Context* profiles relevant to their area of business. Users within the community do not create,
4742 define or manage their individual *Context* needs but conform to the community’s standard. Such a
4743 community should liase closely with other communities and with general standards-making bodies to
4744 avoid overlapping work. A community may be as small as two consenting organisations.

4745 *Version* – An indication of the evolution over time of an instance of a *Core Component*, *Data Type*,
4746 *Business Context*, or *Business Information Entity*.

4747 *XML schema* – A generic term used to identify the family of grammar based XML document structure
4748 validation languages to include the more formal W3C XML Schema Technical Specification, Document
4749 Type Definition, Schematron, Regular Language Description for XML (RELAX), and the OASIS RELAX
4750 NG.

4751

4752

4753 **Appendix J. Qualified Data Type Schema Module**

4754 The Qualified Data Type Schema Module is published as a separate file – QualifiedDataType.xsd.

4755

4756

Copyright Statement

Copyright © UN/CEFACT 2005. All Rights Reserved.

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

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

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