UBL Naming and Design Rules Version 3.0

Committee Specification Draft 01 / Public Review Draft 01

4 February 2015

Specification URIs

This version:

http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/csprd01/UBL-NDR-v3.0-csprd01.xml (Authoritative)
http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/csprd01/UBL-NDR-v3.0-csprd01.html
http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/csprd01/UBL-NDR-v3.0-csprd01.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/UBL-NDR-v3.0.html
http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/UBL-NDR-v3.0.pdf

Technical Committee:

OASIS Universal Business Language (UBL) TC

Chairs:

Tim McGrath (tim.mcgrath@documentengineeringservices.com), Document Engineering Services Limited
G. Ken Holman (gkholman@CraneSoftwrights.com), Crane Softwrights Ltd.

Editor:

G. Ken Holman (gkholman@CraneSoftwrights.com), Crane Softwrights Ltd.

Related work:

This specification replaces or supersedes:

Universal Business Language (UBL) 2.0 Naming and Design Rules. Edited by Mike Grimley and Mavis Cournane. 25 December 2009. Committee Specification 01. http://docs.oasis-open.org/ubl/UBL-2.0-NDR/UBL-2.0-NDR.html.

This specification is related to:

Universal Business Language Version 2.1. Edited by Jon Bosak, Tim McGrath and G. Ken Holman. 04 November 2013. OASIS Standard. http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html.

Abstract:

This specification prescribes a set of naming and design rules used to create XML document model validation artefacts (W3C XSD schemas and OASIS Context/value association files) associated with abstract information bundles formally described using the ISO/TS 15000-5:2005 Core Component

Technical Specification [**CCTS - ISO/TS 15000-5:2005**]. These rules are not limited only for use with UBL.

This work product also describes how, in particular, these rules were followed to create the information bundles and XML document model validation artefacts for UBL [**UBL-2.1**] documents and extensions starting with UBL version 2.1.

These rules can be followed to create any or all of the following:

- custom extensions for use with UBL documents; and
- custom standalone documents based on the UBL common library.

See Appendix A, *Release notes (Non-Normative)* for more information regarding this release package.

Citation format:

When referencing this specification the following citation format should be used:

[**UBL-NDR-3.0**] *UBL Naming and Design Rules Version 3.0.* Edited by G. Ken Holman. 4 February 2015. Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/csprd01/UBL-NDR-v3.0-csprd01.html. Latest version: http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/UBL-NDR-v3.0.html.

Table of Contents

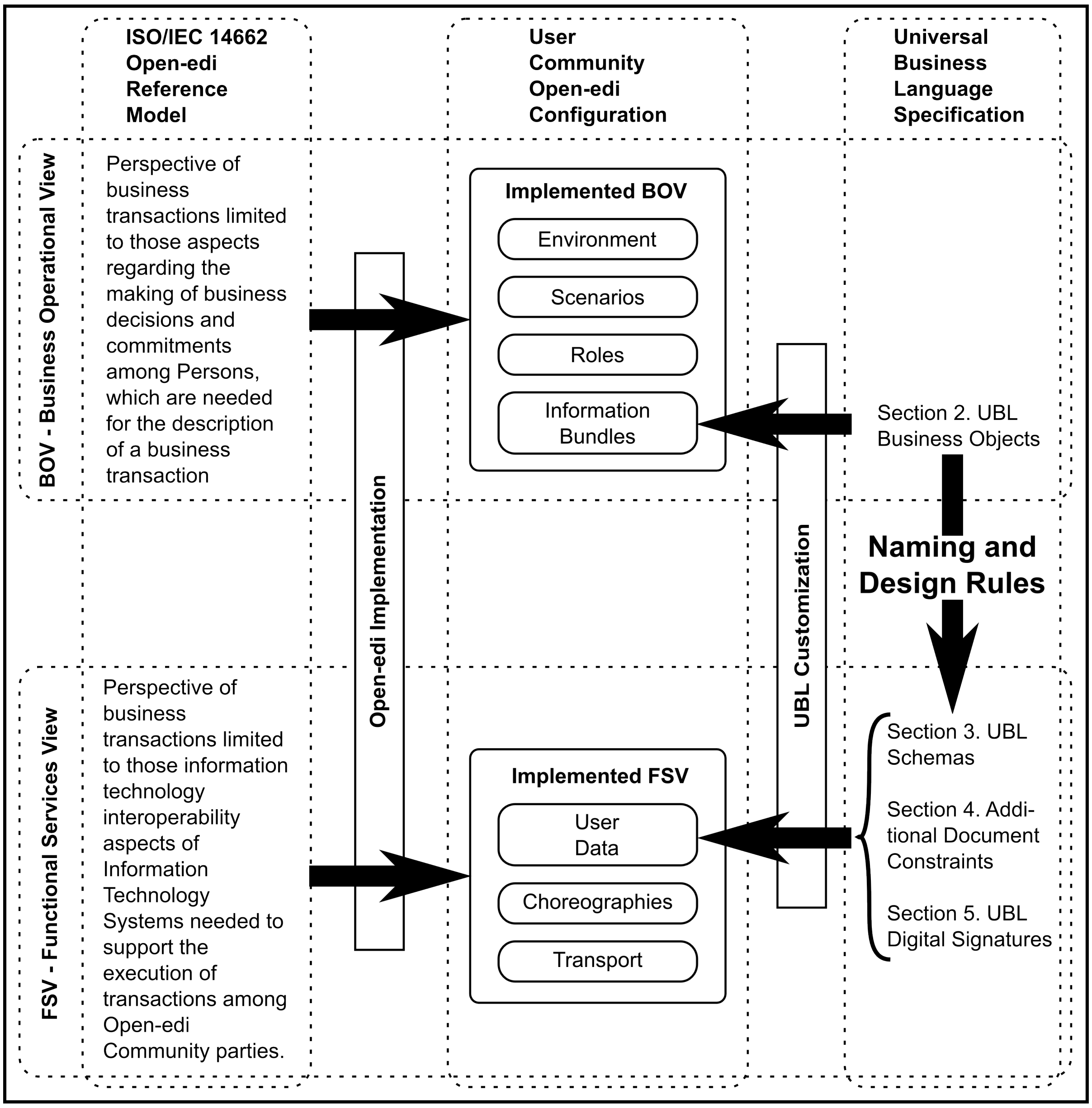
1 Introduction

The Open-edi Reference Model [ISO 14662] distinguishes an information bundle as the abstract description of the structure and semantics of the information exchanged between parties, separate from the user data syntax of the exchange of that information.

The use of the user data syntax in that exchange can be constrained by validation artefacts derived from the information bundles. A validation process can then report violations of the syntax constraints.

Naming and Design Rules (NDR) describe the process by which the user data syntax constraint mechanisms are created from the abstract definition of the information bundles. This is illustrated in Figure 1, “Open-edi Application”, reproduced from OASIS Standard UBL 2.1 Figure H.2 with an annotation highlighting the relationship of the NDR applied to UBL. The original figure depicts the roles of the four normative sections of the UBL 2.1 specification.

Figure 1. Open-edi Application



ISO/IEC 14662 Open-edi Reference Model
User Community Open-edi Configuration
Universal Business Language Specification
BOV - Business Operational View
Perspective of business transactions limited to those aspects regarding the making of business decisions and commitments among Persons, which are needed for the description of a business transaction
Open-edi Implementation
Implemented BOV
Environment
Scenarios
Roles
Information Bundles
UBL Customization
Section 2. UBL Business Objects
Naming and Design Rules
FSV - Functional Services View
Perspective of business transactions limited to those information technology interoperability aspects of Information Technology Systems needed to support the execution of transactions among Open-edi Community parties.
Implemented FSV
User Data
Choreographies
Transport
Section 3. UBL Schemas
Section 4. Additional Document Constraints
Section 5. UBL Digital Signatures

This work product fulfills a commitment by the UBL Technical Committee to document the maintenance of the information bundles and the production of the XML document model validation artefacts for UBL 2.1 [**UBL-2.1**] documents expressing user data. The UBL information bundles are described in the abstract using CCTS [**CCTS - ISO/TS 15000-5:2005**]. The UBL XML user data document model validation artefacts are described using W3C XML Schema [XSD schema] and OASIS Context/value association [**CVA**] expressions.

This work product has been written primarily as a normative prescriptive document, including a conformance section, for those designing new information bundles for new validation artefacts for any business vocabulary, not just UBL.

Part of this work product is a non-normative descriptive document for reference regarding the application of these maintenance and production practices as applied by the UBL Technical Committee when creating UBL 2.1.

Note

Until these rules change, further releases of UBL will continue to be created according to these rules, but the non-normative sections of this document will not be revised to reflect any new release.

For readers working with UBL either for use with the UBL validation artefacts, or for standalone use these rules provide guidance when writing custom extensions for UBL document models and when writing custom documents using the UBL common library.

To achieve these two prescriptive and descriptive goals, this work product expresses a set of naming and design rules used to create and use tools to express the information bundle of an exchange between parties, and then create XML document model validation artefacts (XSD schemas and context/value association files) from those tools.

Note

The UBL 2.1 specification is backwards compatible with its predecessor UBL 2.0. Both specifications were completed before these Naming and Design Rules were formalized. There may very well be in those specifications some isolated violations of these rules. Changing the specifications to remove these violations is not possible because of the commitment to maintain backward compatibility with established implementations of the published specifications. Such rule violations do not impact on the integrity of the specifications, as these rules represent a best practice promoted by the UBL Technical Committee.

These rules presume the reader is already familiar with the following and this document makes no attempt to reiterate any or all definitions, principles, rules, syntax or constraints found in the following:

- the principles of ISO/TS 15000-5:2005 Core Component Technical Specification [**CCTS - ISO/TS 15000-5:2005**];
- the principles of ISO/IEC 11179 Data Elements [**ISO 11179**];
- the semantics and syntax of W3C XSD [XSD schema] for XML document constraint specification;
- the semantics and syntax of OASIS Context/value association using genericode [**CVA**] for code list association; and
- the semantics and syntax of OASIS genericode [**genericode**] for code list representation.

Naming and design rules govern the application of the above works to the objective of creating concrete user data validation artefacts from abstract information bundles.

Documentation conventions

The documentation convention followed in the major normative clauses of this specification details the normative prescriptive rules in the first subclause and describe the non-normative UBL 2.1 application of those rules in the second subclause. Where possible, subsequent sub-subclauses under each subclause correspond the NDR rules with the UBL 2.1 application of those rules.

1.1 Terminology

1.1.1 Key words

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* in this document are to be interpreted as described in [**RFC 2119**]. Note that for reasons of style, these words are not capitalized in this document.

1.1.2 Terms and Definitions

Context/value Association
: The association of value constraints imposed on information found in a particular document context.

Core Component Technical Specification
: A methodology for creating a business vocabulary that can be expressed in one or more syntactic expressions, as defined in [**CCTS - ISO/TS 15000-5:2005**].

document ABIE
: The apex ABIE of an information bundle.

document element
: The apex element of information in an XML document, as defined in [**XML**].

information bundle
: The formal description of the semantics of the recorded information to be exchanged, as defined in [**ISO 14662**].

naming and design rules
: A set of rules governing the expression of information bundles using Core Component Technical Specification, and the creation of associated XML document model validation artefacts using Context/value Association and XSD schema.

object class
: A set of ideas, abstractions, or things in the real world that are identified with explicit boundaries and meaning and whose properties and behavior follow the same rules [**ISO 11179**] (definition 3.3.22).

property
: A characteristic common to all members of an object class [**ISO 11179**] (definition 3.3.29).

XSD schema
: An XML document model definition conforming to the W3C XML Schema language [**XSD1**][**XSD2**].

The terms *Core Component (CC), Basic Core Component (BCC), Aggregate Core Component (ACC), Association Core Component (ASCC), Business Information Entity (BIE), Basic Business Information Entity (BBIE),* and *Aggregate Business Information Entity (ABIE)* are used in this specification with the meanings given in [**CCTS - ISO/TS 15000-5:2005**].

The terms *Object Class, Property Term, Representation Term,* and *Qualifier* are used in this specification with the meanings given in [**ISO 11179**].

1.1.3 Symbols and Abbreviations

ABIE
: Aggregate Business Information Entity [**CCTS - ISO/TS 15000-5:2005**]

ASBIE
: Association Business Information Entity [**CCTS - ISO/TS 15000-5:2005**]

BIE
: Business Information Entity [**CCTS - ISO/TS 15000-5:2005**]

BBIE
: Basic Business Information Entity [**CCTS - ISO/TS 15000-5:2005**]

CCTS
: Core Component Technical Specification

CVA
: Context/value Association

NDR
: naming and design rules

TC
: Technical Committee

XSD
: W3C XML Schema Definition [XSD schema]

1.1.4 Key concepts

Extension collection
: The set of extension items found in an XML document, constrained by an extension schema, that supplements the base information that is constrained by the published document model.

Extension item
: A single instance of structured supplemental information and its associated metadata distinguishing it from other extension items.

Extension point
: The apex element of structured supplemental information described by its metadata.

XML document validation
: The processing involved in confirming that an XML document satisfies the document constraints expressed in a set of artefacts governing aspects of document content according to a document model.

1.2 Normative References

[**CCTS - ISO/TS 15000-5:2005**] *ISO/TS 15000-5:2005 Electronic Business Extensible Markup Language (ebXML)— Part 5: ebXML Core Components Technical Specification, Version 2.01 [https://www.oasis-open.org/committees/download.php/6232/CEFACT-CCTS-Version-2pt01.zip] (identical to Part 8 of the ebXML Framework)*

[**CVA**] *OASIS Context/value association using genericode 1.0.* 15 April 2010. Committee Specification 01. http://docs.oasis-open.org/codelist/cs01-ContextValueAssociation-1.0/doc/context-value-association.html.

[**genericode**] *OASIS Code List Representation (Genericode) Version 1.0.* 28 December 2007. Committee Specification 01. http://docs.oasis-open.org/codelist/cs-genericode-1.0/doc/oasis-code-list-representation-genericode.html.

[**ISO 14662**] *ISO/IEC 14662:2004 Information technology — Open-edi reference model [http://standards.iso.org/ittf/PubliclyAvailableStandards/]*

[**RFC 2119**] *Key words for use in RFCs to Indicate Requirement Levels*, March 1997. S. Bradner. IETF (Internet Engineering Task Force) RFC 2119, *http://www.ietf.org/rfc/rfc2119.txt*

[**XML**] *Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008 [http://www.w3.org/TR/2008/REC-xml-20081126/]*

[**XPath 1.0**] James Clark, Steve DeRose, *XML Path Language (XPath) Version 1.0, 16 November 1999 [http://www.w3.org/TR/1999/REC-xpath-19991116/].*

[**XSD1**] *XML Schema Part 1: Structures. Second Edition. W3C Recommendation 28 October 2004 [http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/]*

[**XSD2**] *XML Schema Part 2: Datatypes. Second Edition. W3C Recommendation 28 October 2004 [http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/]*

1.3 Non-Normative References

[**ISO 11179**] *ISO/IEC 11179-1:2004 Information technology — Specification and standardization of data elements — Part 1: Framework for the specification and standardization of data elements [http://www.oasis-open.org/committees/download.php/6233/c002349_ISO_IEC_11179-1_1999%28E%29.pdf]*

[**UBL-2.1**] *Universal Business Language Version 2.1.* 4 February 2015. Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html.

[**UBL-CCTS**] *UBL Conformance to ebXML CCTS ISO/TS 15000-5:2005 Version 1.0* 4 February 2015. Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/ubl/UBL-conformance-to-CCTS/v1.0/.

[**xmldsig**] *XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002 [http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/]*

2 Context of use and application of these rules

Naming and design rules govern the production of user data validation artefacts from the structure of information bundles describing the content of exchanges between parties as described in the Open-edi Reference Model [**ISO 14662**].

This standard mandates that for a given suite of document types, all information bundles be organized and maintained in such a way as to create the user data validation artefacts for that information using a well-defined process.

The information bundle is expressed in terms and concepts of the Core Component Technical Specification. The user data validation artefacts for a document model are twofold: the document structural constraints (e.g. nesting and cardinality) of elements and attributes in XML documents [**XML**], and the data type qualifications (e.g. coded value domains, a.k.a. code lists). Document structural constraints are expressed using XSD schema semantics and syntax XSD schema. Data type qualifications are expressed using CVA associations Context/value Association of code list values [**genericode**] with XML document locations using XPath [**XPath 1.0**].

2.1 Generic process overview

This specification presents this process of producing user data validation artefacts as having at least four steps, two of which are always present and at least two, among possible others, that are optional.

There shall be a process by which CCTS information regarding the information bundles is managed. This may be in a collaboration tool of some kind so as to involve multiple participants. The tool itself may be as simple as a file being shared back and forth between participants (or maintained solely by an individual if there are no other participants), or as complex as an online interactive browser-based tool with credentialed access offering information bundle visualization.

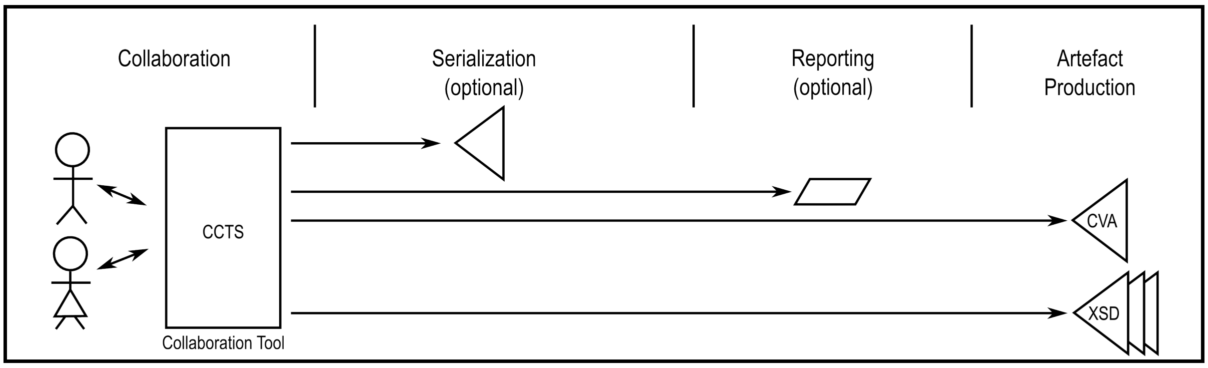
GEN01 Maintenance of information bundle composition
There shall be a documented process by which information bundles are managed. Such a process shall support principles of the Core Component Technical Specification as described in other rules in this specification. **Note** While not mandatory, such a process should support collaborative interaction among project members responsible for the information being maintained.

There shall be a process in which user data validation artefacts are produced. These artefacts reflect the information bundles as constraints on the user data syntax used to represent instances being exchanged. These may be produced directly from the collaboration tool or indirectly from intermediate files created from the collaboration tool.

GEN02 Production of validation artefacts
There shall be a documented process by which user data validation artefacts are created from the information maintained for information bundles.

There may be any number of optional processes to produce additional artefacts may be useful, two of which are depicted in Figure 2, “Generic NDR process to create validation artefacts”: the serialization of the CCTS information into a form suitable for additional processing, and the production of reports useful in the collaboration and review of the work.

Figure 2. Generic NDR process to create validation artefacts



2.2 UBL process overview (Non-Normative)

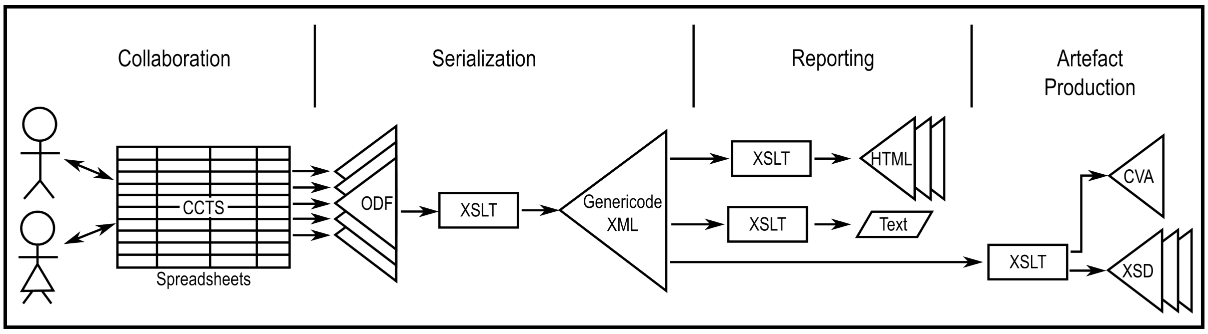
The processes to produce artefacts engaged by the UBL TC are depicted in Figure 3, "UBL NDR process to create validation artefacts".

Office spreadsheets are used to maintain CCTS information describing information bundles in a manner suitable for collaboration amongst members of the technical committee. This information is exported in an XML vocabulary using OASIS Open Document Format. This is then merged using XSLT to produce a single serialization of the CCTS information in XML using the OASIS genericode vocabulary.

The genericode file is then used to produce both an HTML summary of all of the models and a model checking report detailing problems with the CCTS information. These reporting artefacts are used by the collaborators to polish the CCTS information.

The genericode file is also used to create the validation artefacts.

Figure 3. UBL NDR process to create validation artefacts



3 Documents, libraries, components and extensions

These naming and design rules provide for creating a collection of extensible documents. Each document is an information bundle of business information entities sharing one or more common libraries of business information entities. Each business information entity is referred to as a component of the information bundle of the document or library. Each component corresponds to an XML element.

A document extension provides for content that augments an information bundle with additional information not previously defined for the information bundle. Extensions, themselves, can contain components and can reference components. Extensions can also include foreign constructs defined by other organizations not using CCTS for structuring information. An extension can have horizontal applicability to all documents, such as the scaffolding to house a digital signature. An extension can have vertical applicability to only a single document, such as additional invoice line detail information to augment an invoice for a specific industry.

Each exchange between partners is realized as a machine-readable instance of the information bundle and its use of any common libraries. When using XML for exchange, the XML document presents the information bundle as a serialization of components, each component being an XML element. Extensions in XML documents provide for including XML elements of components or of other XML vocabularies.

3.1 Rules for structuring information bundles

An information bundle is described as a collection of components. Per CCTS, a component's type is either an Aggregate Business Information Entity (ABIE), an Association Business Information Entity (ASBIE) or a Basic Business Information Entity (BBIE).

3.1.1 Documents

A document ABIE structures the apex of the information bundle to be exchanged between parties.

MOD01 Information bundle structured as a document ABIE
Each document shall be structured as a single top-level ABIE, referred to in this specification as a *document ABIE*. **Note** The rationale is that the document ABIE is always considered a one-member collection in and of itself with no other members in the collection. The collection is identified in syntax separately from other collections of ABIEs.

3.1.2 Libraries

A library is a collection of shared ABIEs that does not include a document ABIE. There may be one or more libraries.

MOD02 Libraries of available ABIEs
Each library shall be structured as a collection of one or more ABIEs. Every collection shall not contain a document ABIE.

MOD02 Libraries of available ABIEs
Note The rationale is that each library collection is identified in syntax distinguished from other collections of ABIEs.

MOD03 Library ordering
The ABIEs in a library shall be arranged in alphabetical order of the ABIE's dictionary entry name. **Note** The rationale is that libraries can be very large and a reader new to the library may be unfamiliar with any arbitrary semantic ordering of the ABIE components. The collection of ABIEs can be navigated more easily when in alphabetical order.

3.1.3 Components

By CCTS definition each component business information entity is based on a core component, either implicitly inferred or explicitly identified.

These rules do not mandate that a BIE's core component be explicitly identified if the designers accept that the BIE's core component is implicitly inferred.

3.1.4 Extensions

By definition an extension is information that cannot otherwise be expressed using the components of a document or library. Wherever possible information should be expressed using the components found in a document library. Situations exist, however, where supplementary information must be added to the information bundle in a way that does not interfere with existing components.

Multiple extensions may apply for any given document information bundle. Extensions may be horizontal in nature, available to use for all documents. Extensions may be vertical in nature, applicable only to a single document.

The extension collection provides a home for the extensions to be used with a document information bundle. Within each collection there may be a number of extension items, each with extension metadata regarding the item and each with a single extension point at which is placed the apex element of the supplemental information.

MOD04 Extension collection in each document information bundle
Each information bundle shall allow for optional augmentation of a document with a collection of information not conceptually modeled by the document ABIE or the library ABIEs. There are no constraints on the information that may be included in an extension, including information concepts built using existing library components.

3.2 UBL structuring of information bundles (Non-Normative)

3.2.1 UBL Documents

In UBL 2.1 no document-level ABIE is referenced by an ASBIE. This merely reflects user requirements to date and in a future revision of UBL there may be such references.

There are 65 document ABIEs representing information bundles in UBL 2.1. The class names for these ABIEs are as follows:

Table 1. UBL 2.1 Document ABIEs

• Application Response • Attached Document • Awarded Notification • Bill Of Lading • Call For Tenders • Catalogue • Catalogue Deletion • Catalogue Item Specification Update • Catalogue Pricing Update • Catalogue Request • Certificate Of Origin • Contract Award Notice • Contract Notice • Credit Note • Debit Note • Despatch Advice • Document Status • Document Status Request • Exception Criteria • Exception Notification • Forecast • Forecast Revision	• Forwarding Instructions • Freight Invoice • Fulfilment Cancellation • Goods Item Itinerary • Guarantee Certificate • Instruction For Returns • Inventory Report • Invoice • Item Information Request • Order • Order Cancellation • Order Change • Order Response • Order Response Simple • Packing List • Prior Information Notice • Product Activity • Quotation • Receipt Advice • Reminder • Remittance Advice • Request For Quotation	• Retail Event • Self Billed Credit Note • Self Billed Invoice • Statement • Stock Availability Report • Tender • Tenderer Qualification • Tenderer Qualification Response • Tender Receipt • Trade Item Location Profile • Transportation Status • Transportation Status Request • Transport Execution Plan • Transport Execution Plan Request • Transport Progress Status • Transport Progress Status Request • Transport Service Description • Transport Service Description Request • Unawarded Notification • Utility Statement • Waybill

3.2.2 UBL Libraries

In UBL 2.1 there is one collection of library ABIEs:

- a common library of 228 ABIEs for reference by ASBIEs in the document ABIEs and extension ABIEs

3.2.3 UBL Components

The core components upon which UBL BIEs are based are not instantiated or published. Only the BIEs that are based upon them are published. This is described in detail in [**UBL-CCTS**]

3.2.4 UBL Extensions

UBL provides an extension collection for every document type. Using the collection is optional and when it exists it contains one or more extensions. For each extension there is provision for a number of extension metadata information items defined by UBL in addition to the extension point itself and its content.

In UBL 2.1 there is one extension provided, that being for digital signatures. The information in this extension is modeled primarily using CCTS but with an additional data item modeled by the W3C in [**xmldsig**]. The components described using CCTS are in a single collection of two ABIEs describing the signature extension ABIE and another ABIE for reference by the extension's ASBIEs.

4 Component descriptions

A component in the information bundle model corresponds to an element in the XML document.

A single component in the document models is described by a number of values. Some of these values are available to be defined by the data modelers, and some of those are optional while others are mandatory. Those values of the description not available to be defined by the data modeler are derived from the definitions of other values which may be entered or, themselves, derived.

The derived values are mandatory but some tools may not provide for their dynamic evaluation into the target value. Where such control is not available and these values are entered manually, a verification process should be run on the values to ensure they were entered without error. Another option would be to prevent the manual entry of these items and require the extraction process to export the derived values.

4.1 Rules for describing components

Note

When reading these rules, be sure to distinguish the edited fields that contribute to the component's dictionary entry name from the one derived field that is the component's name to be used in user data.

4.1.1 Recorded CCTS information

These rules govern the creation and expression of values found in the CCTS information model describing the information bundle.

COM01 CCTS information item description values
Each item of describing information shall be a string value of Unicode characters without embedded hierarchical structure. The value itself may be structured in its syntax within the string. **Note** The rationale is that downstream expressions of the model information may be constrained in its expression, such as is true in an XML attribute.

COM02 CCTS information item prohibited characters and sequence
All items of describing information shall be void of all sensitive XML markup characters and sequences. The following characters are prohibited in any CCTS information item: • the characters "<", ">", "&" • the character sequence "--" **Note** The rationale is that prohibiting these characters and sequence will allow the information to be processed more simply in different XML contexts without special handling.

4.1.2 Abbreviations

It may be convenient to abbreviate often-repeated words or to use industry-accepted abbreviations in CCTS information.

COM03 Agreement on a list and rules of abbreviation in names

The users of a CCTS model shall agree on a list (that may be empty) of words found in naming values that shall be abbreviated when a word is used in the component's derived name.

Examples

"Identifier" is abbreviated as "ID".
"Universally Unique Identifier" is abbreviated as "UUID".

Note

The rationale is that some common terms have widely-accepted abbreviations suitable for shortening derived names.

COM04 Agreement on a list and rules of abbreviation in model values

The users of a CCTS model shall agree on a list (that may be empty) of accepted abbreviations allowed in values.

The only abbreviations that shall be used in values are those found in the agreed-upon list.

Examples

"XML Path Language" is abbreviated as "XPath".
"Card Verification Value" is abbreviated as "CV2".

Note

The rationale is that some common terms have widely-accepted abbreviations in general use or in particular use within the information domain. Inconsistent use of abbreviations may lead to confusion by users.

COM05 Agreement on a list of equivalences in names

The users of a CCTS model shall agree on a list (that may be empty) of words found in naming values that shall be considered equivalent when a word is used in the component's derived name.

Example

The property term primary noun "URI" is considered equivalent to the representation term "Identifier".

Note

The rationale is that some common terms wholly include the concepts presented in other terms and so should be considered equivalent in order to prevent duplication.

COM06 Leading name part in attribute names
Every attribute's derived name shall be composed with the leading name part entirely in lower-case, even when that name part is an agreed-upon abbreviation. **Example** "Numeric. Format. Text" is represented with the attribute named "format" **Note** Words in attribute names of supplementary components of CCTS Table 8-2 "Approved Core Component Type Content and Supplementary Components" are subject to abbreviation.

COM07 Non-leading abbreviations in attribute names
When an attribute's derived name is composed with an agreed-upon abbreviation in other than the leading name part, the abbreviation shall be entirely in upper-case. **Example** "Amount Currency. Identifier" is represented with the attribute named "currency-ID" "Amount Currency. Code List Version. Identifier" is represented with the attribute named "currencyCodeListVersionID" **Note** Words in attribute names of supplementary components of CCTS Table 8-2 "Approved Core Component Type Content and Supplementary Components" are subject to abbreviation.

4.1.3 CCTS name value conventions

COM08 CCTS information item name value prohibited characters
All information items contributing to a component's dictionary entry name shall be void of all sensitive dictionary entry name markup characters. The following characters are prohibited in any CCTS information item that contributes to the dictionary entry name: • the characters " " (space), "." (period) and "_" (underscore) **Note** The rationale is that prohibiting these characters in name values prevents ambiguity when assembled into the dictionary entry name using these characters to provide structure.

COM09 Use of the singular form in names
All information items contributing to a component's dictionary entry name shall be in the singular form unless the concept itself is plural.

COM09 Use of the singular form in names

Example

"Payment Terms" is the noun used for the collection of a plurality of various aspects of payment and has a cardinality of only "0..1".

Note

The rationale is that the cardinality of an item named in the singular confers the plurality of the item.

COM10 Use of leading upper case letter in name value items

All words that are not abbreviated in name values contributing to the dictionary entry name shall have a leading upper-case letter and all other letters in lower-case.

4.1.4 BIE definition conventions

These NDR require a minimum set of information to be recorded for each type of BIE.

Note

The "Name" value in BIE definitions may also be labeled the "Component Name" value or some other characteristic label provided such label does not conflict with any other value's label.

COM11 Minimum set of values describing an ABIE component

Each ABIE component of the model shall have at least a prescribed set of values required or available for definition.

This set is as follows:

- Component Type (fixed)
  - shall be the value "ABIE"
- Definition (mandatory)
  - this value identifies the component using complete natural language sentences in a single paragraph
- Alternative Business Terms (optional)
  - this value lists other natural language terminology for the component that is acceptable in common use
- Object Class Qualifier (optional)
  - when used this value distinguishes one object class from a like-named other
- Object Class (mandatory)
  - this value identifies the object class as a collection of BBIE and ASBIE components
- Name (mandatory; derived)
  - this value shall be the concatenation of Object Class Qualifier and the Object Class without any spaces, abbreviating the values as defined

COM11 Minimum set of values describing an ABIE component

Example formula

Given the following:

$OCQ = Object Class Qualifier
$OC = Object Class

=SUBSTITUTE(CONCATENATE($OCQ;$OC);" ";"")

- Dictionary Entry Name (mandatory; derived)
  - this value shall be the concatenation of the Object Class Qualifier, followed by an underscore and space when defined, followed by the Object Class, followed by a period and space, followed by the word "Details"

Example formula

Given the following:

$OCQ = Object Class Qualifier
$OC = Object Class

=CONCATENATE(IF($OCQ="";"";CONCATENATE($OCQ;"_ "));$OC;". Details")

Example ABIE

Fixed value:

Component Type="ABIE"

Entered values:

Definition="A class to define common information related to an address."
Object Class="Address"

Derived values:

Name="Address"
Dictionary Entry Name="Address. Details"

COM12 Minimum set of values describing a BBIE component

Each BBIE component of the model shall have at least a prescribed set of values required or available for definition.

This set is as follows:

- Component Type (fixed)
  - shall be the value "BBIE"
- Cardinality (mandatory)
  - one of "1" (required and not repeatable), "0..1" (optional and not repeatable), "0..n" (optional and repeatable) and "1..n" (required and repeatable)
- Definition (mandatory)

COM12 Minimum set of values describing a BBIE component

- this value identifies the component using complete natural language sentences in a single paragraph
- Alternative Business Terms (optional)
  - this value lists other natural language terminology for the component that is acceptable in common use
- Object Class Qualifier (optional)
  - when used this value distinguishes one object class from a like-named other
- Object Class (mandatory)
  - this value identifies the object class of the ABIE to which the BBIE belongs
- Property Term Qualifier (optional)
  - when used this value distinguishes one property term from a like-named other
- Property Term Possessive Noun (optional)
  - when used this value identifies a characteristic of the nature of the component
- Property Term Primary Noun (mandatory)
  - this value identifies the principle nature of the component
- Representation Term (mandatory)
  - this value identifies the form of the instantiated component value and shall be selected from the set of 20 primary and secondary representation terms itemized in CCTS Table 8.3 Permissible Representation Terms (ordered by primary term with secondary terms in parentheses):
    - Amount
    - Binary Object (Graphic, Picture, Sound, Video)
    - Code
    - Date Time (Date, Time)
    - Identifier
    - Indicator
    - Measure
    - Numeric (Value, Rate, Percent)
    - Quantity
    - Text (Name)
- Data Type Qualifier (optional)
  - when used this value distinguishes particular restrictions on a data type from the use of a data type with other (or no) restrictions

COM12 Minimum set of values describing a BBIE component

- Name (mandatory; derived)
  - this value shall be the concatenation of Property Term Qualifier, Property Term Possessive Noun and, when the Property Term Primary Noun is not "Text" or it is "Text" and both the Property Term Qualifier and the Property Term Possessive Noun are not defined, then the Property Term Primary Noun (abbreviating it as required) and, when the Representation Term is not "Text" and the Property Term Primary Noun is not equivalent to the Representation Term, then also the Representation Term component (abbreviating it as required), all without intervening spaces

  ### Example formula

  Given the following:

  $PTQ = Property Term Qualifier
  $PTPsN = Property Term Possessive Noun
  $PTPrN = Property Term Primary Noun
  $RT = Representation Term

  ```
  =SUBSTITUTE( CONCATENATE( $PTQ;$PTPsN;IF( $PTPrN="Identifi-
  er";"ID"; IF( AND( $PTPrN="Text";OR( $PTQ<>"";$PTPsN<>""));"";$PTPrN));
  IF( AND( $RT<>"Text";$PTPrN<>$RT;NOT( AND( $PT-
  PrN="URI";$RT="Identifier")); NOT( AND( $PTPrN="UUID";$RT="Identifi-
  er")));IF( $RT="Identifier";"ID";$RT);""));" ";"")
  ```

- Dictionary Entry Name (mandatory; derived)
  - this value shall be the concatenation of the Object Class Qualifier, followed by an underscore and space when defined, followed by the Object Class, followed by a period and space, followed by the Property Term Qualifier, followed by an underscore and space when defined, followed by the Property Term, and then, when either the Property Term Qualifier is defined or the Property Term is not equal to the Representation Term, followed by a period and space and the Representation Term

  ### Example formula

  Given the following:

  $OCQ = Object Class Qualifier
  $OC = Object Class
  $PTQ = Property Term Qualifier
  $PT = Property Term
  $RT = Representation Term

  ```
  =CONCATENATE( IF( $OCQ="";"";CONCATENATE( $OCQ;"_ "));$OC;".
  ";IF( $PTQ="";"";CONCATENATE( $PTQ;"_ "));$PT;IF( OR(
  $PTQ<>"";$PT<>$RT);CONCATENATE( ". ";$RT);""))
  ```

- Property Term (mandatory; derived)
  - this value shall be the concatenation of Property Term Possessive Noun followed by a space should it exist, followed by the Property Term Primary Noun

COM12 Minimum set of values describing a BBIE component

Example formula

Given the following:

$PTPsN = Property Term Possessive Noun
$PTPrN = Property Term Primary Noun

=IF($PTPsN<>"";CONCATENATE($PTPsN;" ";$PTPrN);$PTPrN)

- Data Type (mandatory; derived)
  - this value shall be the concatenation of the Data Type Qualifier followed by an underscore and space when it exists, the Representation Term, followed by a period and space, followed by the word "Type"

Example formula

Given the following:

$DTQ = Data Type Qualifier
$RT = Representation Term

=CONCATENATE(IF($DTQ="";"";CONCATENATE($DTQ;"_ "));$RT;". Type")

Example BBIE

Fixed value:

Component Type="BBIE"

Entered values:

Cardinality="0..1"
Definition="An additional street name used to further clarify the address."
Object Class="Address"
Property Term Qualifier="Additional"
Property Term Possessive Noun="Street"
Property Term Primary Noun="Name"
Representation Term="Name"

Derived values:

Name="AdditionalStreetName"
Dictionary Entry Name="Address. Additional_ Street Name. Name"
Property Term="Street Name"
Data Type="Name. Type"

COM13 Minimum set of values describing an ASBIE component

Each ASBIE component of the model shall have at least a prescribed set of values required or available for definition.

This set is as follows:

- Component Type (fixed)
  - shall be the value "ASBIE"

COM13 Minimum set of values describing an ASBIE component

- Cardinality (mandatory)
  - one of "1" (required and not repeatable), "0..1" (optional and not repeatable), "0..n" (optional and repeatable) and "1..n" (required and repeatable)
- Definition (mandatory)
  - this value identifies the component using complete natural language sentences in a single paragraph
- Alternative Business Terms (optional)
  - this value lists other natural language terminology for the component that is acceptable in common use
- Examples (optional)
  - this is a free-form value used to illustrate the definition
- Object Class Qualifier (optional)
  - when used this value distinguishes one object class from a like-named other
- Object Class (mandatory)
  - this value identifies the object class of the ABIE to which the ASBIE belongs
- Associated Object Class Qualifier (optional)
  - when used this value distinguishes one object class from a like-named other
- Associated Object Class (mandatory)
  - this value identifies the object class of the ABIE to which the ASBIE belongs
- Property Term Qualifier (optional)
  - when used this value distinguishes one property term from a like-named other
- Name (mandatory; derived)
  - this value shall be the concatenation of Property Term Qualifier and the Property Term without any spaces or underscore, abbreviating the values as defined

    ### Example formula

    Given the following:

    $PTQ = Property Term Qualifier
    $PT = Property Term

    =SUBSTITUTE(SUBSTITUTE(CONCATENATE($PTQ;IF($PT="Identifier";"ID";$PT));" ";"");"_";"")
- Dictionary Entry Name (mandatory; derived)
  - this value shall be the concatenation of the Object Class Qualifier, followed by an underscore and space when defined, followed by the Object Class, followed by a period and space, followed by the Property Term Qualifier, followed by and

COM13 Minimum set of values describing an ASBIE component

underscore and space when defined, followed by the Property Term, and then when the Property Term Qualifier is defined, followed by a period and space and the Representation Term

Example formula

Given the following:

$OCQ = Object Class Qualifier
$OC = Object Class
$PTQ = Property Term Qualifier
$PT = Property Term
$RT = Representation Term

=CONCATENATE(IF($OCQ="";"";CONCATENATE($OCQ;"_ "));$OC;".
";IF($PTQ="";"";CONCATENATE($PTQ;"_ "));$PT;IF($PTQ="";"";CON-
CATENATE(". ";$RT)))

- Property Term (mandatory; derived)
  - this value shall be the concatenation of the Associated Object Class Qualifier, an underscore and a space when defined, and the Associated Object Class

Example formula

Given the following:

$AOCQ = Associated Object Class Qualifier
$AOC = Associated Object Class

=CONCATENATE(IF($AOCQ="";"";CONCATENATE($AOCQ;"_ "));$AOC)

- Representation Term (mandatory; derived)
  - this value is a copy of the Property Term

Example ASBIE

Fixed value:

Component Type="ASBIE"

Entered values:

Cardinality="0..1"
Definition="The buyer of the item."
Object Class="Forecast"
Associated Object Class="Customer Party"
Property Term Qualifier="Buyer"

Derived values:

Name="BuyerCustomerParty"
Dictionary Entry Name="Forecast. Buyer_ Customer Party. Customer Party"
Property Term="CustomerParty"
Representation Term="CustomerParty"

COM14 ABIE contents cannot be empty
Every ABIE shall contain at least one BIE.

4.2 UBL maintenance of components (Non-Normative)

The UBL TC uses office spreadsheets as the collaborative tool to maintain the CCTS model of the UBL documents, the common library and the signature extension. A spreadsheet row is used for each component. A spreadsheet column is used for each mandatory and optional CCTS facet.

The 65 document ABIEs are described using individual spreadsheets that are found here:

http://docs.oasis-open.org/ubl/os-UBL-2.1/mod/maindoc/

The common library ABIEs are in a single spreadsheet and the signature extension ABIEs are in a single spreadsheet that are found here:

http://docs.oasis-open.org/ubl/os-UBL-2.1/mod/common/

4.2.1 UBL Recorded CCTS information

UBL imposes an additional constraint on the values of recorded CCTS information that all characters used be ASCII characters. This prohibits accented letters, exotic symbols, and stylized punctuation such as "smart quotes". The rationale for this is that the resulting character sequences are then not subject to character encoding vagaries that may not be properly handled by downstream processing applications.

4.2.2 UBL Abbreviations and equivalences

UBL abbreviates the following name value words when used in a component's name:

- "Identifier" is abbreviated as "ID"

UBL considers the following name value words equivalent when used in a component's name:

- the property term "UUID" is equivalent to the representation term "Identifier"
- the property term primary noun "URI" is equivalent to the representation term "Identifier"

UBL allows the use of the following abbreviations in descriptions, name values and in a component's name:

- CV2 is allowed for "Card Verification Value"
- UBL is allowed for "Universal Business Language"
- UNDG is allowed for "United Nations Dangerous Goods"
- URI is allowed for "Uniform Resource Identifier"
- UUID is allowed for "Universally Unique Identifier"
- XPath is allowed for "XML Path Language"

4.2.3 UBL CCTS value conventions

In addition to the constraints mandated by these NDR, the words comprising CCTS information items for a component in UBL that are not abbreviations are in Oxford English.

4.2.4 UBL BIE definition conventions

The CCTS information "Name" value for a BIE definition is labeled "UBL Name" in the UBL spreadsheets.

In addition to the CCTS columns indicated as having to be maintained by these NDR, the UBL spreadsheets contain the following columns of additional information used and recorded during analysis, all of which are optional:

- Examples
- UN/TDED Code
- Current Version
- Analyst Notes
- CCL Dictionary Entry Name
- Context: Business Process
- Context: Region (Geopolitical)
- Context: Official Constraints
- Context: Product
- Context: Industry
- Context Role
- Context: Supporting Role
- Context: System Constraint
- Editor's Notes
- Changes from Previous Version

Object classes are never qualified in UBL. Accordingly, the "Object Class Qualifier" and "Associated Object Class Qualifier" columns are present but always empty in instances of the spreadsheets.

Every Document ABIE contains as its first four BBIE children, in order and all with property Term Primary Noun "Identifier" and Representation Term "Identifier", items with the following Property Term Possessive Nouns:

- UBL Version
- Customization
- Profile
- Profile Execution

5 Optional CCTS serialization

An implementation of these naming and design rules may choose to create a serialization of the CCTS information. This can be a useful convenience when processing the CCTS information as a whole. The CCTS collaboration tool is not required to produce a serialization if such is not needed to fulfill its obligation to produce validation artefacts.

5.1 Rules for CCTS serialization

There are no formal rules for CCTS serialization.

5.2 UBL CCTS serialization (Non-Normative)

The UBL distribution includes the following serializations of CCTS information expressed using the OASIS genericode XML vocabulary:

> http://docs.oasis-open.org/ubl/os-UBL-2.1/mod/
>
> - UBL-Entities-2.1.gc - all document ABIEs and common library ABIEs
> - UBL-Signature-Entities-2.1.gc - all signature extension ABIEs

Note

> The rationale is that while OASIS genericode was designed for use with code lists, it is a suitable XML serialization of any sparsely-populated tabular construct. The table of CCTS information in a model is sparsely-populated with numerous undefined cells.

In addition to a column for each piece of CCTS information saved for each BIE, there is a column named "ModelName" populated by every row. The value in this column indicates the model in which the BIE is defined. The values include "UBL-CommonLibrary-2.1" for all BIEs defined in the common library, and "UBL-XXXX-2.1" where "XXXX" is the component name of the ABIE for the object class (e.g. "FreightInvoice") for the document ABIE and its child BIEs.

The key column in the genericode serialization is declared to be the "DictionaryEntryName" column.

The genericode specification imposes no order on the values in a row. In UBL 2.1 the order happens to correspond to the order of column declarations.

An example serialization of all three kinds of BIE is as follows for the common library ABIE "Winning Party", one of 8 ABIEs in UBL 2.1 that have a single BBIE and a single ASBIE (note that commented numbers indicate the row number from the spreadsheet; this supplemental information is not required to be included but exists as a convenient reference):

```
<Row><!--2354-->
   <Value ColumnRef="ModelName">
      <SimpleValue>UBL-CommonLibrary-2.1</SimpleValue>
   </Value>
   <Value ColumnRef="UBLName">
      <SimpleValue>WinningParty</SimpleValue>
   </Value>
   <Value ColumnRef="DictionaryEntryName">
      <SimpleValue>Winning Party. Details</SimpleValue>
   </Value>
   <Value ColumnRef="ObjectClass">
      <SimpleValue>Winning Party</SimpleValue>
   </Value>
   <Value ColumnRef="ComponentType">
      <SimpleValue>ABIE</SimpleValue>
   </Value>
   <Value ColumnRef="Definition">
      <SimpleValue>A party that is identified as the awarded by a tender
      result.</SimpleValue>
   </Value>
   <Value ColumnRef="CurrentVersion">
      <SimpleValue>2.1</SimpleValue>
   </Value>
   <Value ColumnRef="ContextBusinessProcess">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextRegionGeopolitical">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextOfficialConstraints">
      <SimpleValue>None</SimpleValue>
   </Value>
   <Value ColumnRef="ContextProduct">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextIndustry">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextRole">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextSupportingRole">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextSystemConstraint">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
</Row>
<Row><!--2355-->
   <Value ColumnRef="ModelName">
      <SimpleValue>UBL-CommonLibrary-2.1</SimpleValue>
   </Value>
   <Value ColumnRef="UBLName">
      <SimpleValue>Rank</SimpleValue>
   </Value>
   <Value ColumnRef="DictionaryEntryName">
      <SimpleValue>Winning Party. Rank. Text</SimpleValue>
   </Value>
   <Value ColumnRef="ObjectClass">
      <SimpleValue>Winning Party</SimpleValue>
   </Value>
   <Value ColumnRef="PropertyTermPrimaryNoun">
      <SimpleValue>Rank</SimpleValue>
   </Value>
   <Value ColumnRef="PropertyTerm">
      <SimpleValue>Rank</SimpleValue>
   </Value>
   <Value ColumnRef="RepresentationTerm">
      <SimpleValue>Text</SimpleValue>
   </Value>
   <Value ColumnRef="DataType">
      <SimpleValue>Text. Type</SimpleValue>
   </Value>
   <Value ColumnRef="Cardinality">
      <SimpleValue>0..1</SimpleValue>
   </Value>
   <Value ColumnRef="ComponentType">
      <SimpleValue>BBIE</SimpleValue>
   </Value>
   <Value ColumnRef="Definition">
      <SimpleValue>Indicates the rank obtained in the
      award.</SimpleValue>
   </Value>
   <Value ColumnRef="CurrentVersion">
      <SimpleValue>2.1</SimpleValue>
   </Value>
   <Value ColumnRef="ContextBusinessProcess">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextRegionGeopolitical">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextOfficialConstraints">
      <SimpleValue>None</SimpleValue>
   </Value>
   <Value ColumnRef="ContextProduct">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextIndustry">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextRole">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextSupportingRole">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextSystemConstraint">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
</Row>
<Row><!--2356-->
   <Value ColumnRef="ModelName">
      <SimpleValue>UBL-CommonLibrary-2.1</SimpleValue>
   </Value>
   <Value ColumnRef="UBLName">
      <SimpleValue>Party</SimpleValue>
   </Value>
   <Value ColumnRef="DictionaryEntryName">
      <SimpleValue>Winning Party. Party</SimpleValue>
   </Value>
   <Value ColumnRef="ObjectClass">
      <SimpleValue>Winning Party</SimpleValue>
   </Value>
   <Value ColumnRef="PropertyTerm">
      <SimpleValue>Party</SimpleValue>
   </Value>
   <Value ColumnRef="RepresentationTerm">
      <SimpleValue>Party</SimpleValue>
   </Value>
   <Value ColumnRef="AssociatedObjectClass">
      <SimpleValue>Party</SimpleValue>
   </Value>
   <Value ColumnRef="Cardinality">
      <SimpleValue>1</SimpleValue>
   </Value>
   <Value ColumnRef="ComponentType">
      <SimpleValue>ASBIE</SimpleValue>
   </Value>
   <Value ColumnRef="Definition">
      <SimpleValue>Information about an organization, sub-organization, or
      individual fulfilling a role in a business process.</SimpleValue>
   </Value>
   <Value ColumnRef="CurrentVersion">
      <SimpleValue>2.1</SimpleValue>
   </Value>
   <Value ColumnRef="ContextBusinessProcess">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextRegionGeopolitical">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextOfficialConstraints">
      <SimpleValue>None</SimpleValue>
   </Value>
   <Value ColumnRef="ContextProduct">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextIndustry">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextRole">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextSupportingRole">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
   <Value ColumnRef="ContextSystemConstraint">
      <SimpleValue>In All Contexts</SimpleValue>
   </Value>
</Row>
```

6 Optional reporting

An implementation of these naming and design rules is not required to produce any supplementary reports. These reports can be useful reference materials for review of the CCTS information maintained for the information bundles.

6.1 Rules for reporting

There are no formal rules for reporting.

6.2 UBL reporting (Non-Normative)

6.2.1 HTML summary reports

The UBL distribution includes a number of HTML summary files reporting filtered content from the CCTS information. These files are found in the directory:

http://docs.oasis-open.org/ubl/os-UBL-2.1/mod/summary/

- readme-Reports.html - report descriptions and usage, legend information, and a summary of links to all of the HTML summary files
- http://docs.oasis-open.org/ubl/os-UBL-2.1/mod/summary/reports/ - the subdirectory with all of the reports
    - UBL-AllDocuments-2.1.html - an unfiltered report of the entire CCTS information content
    - UBL-XXXX-2.1.html - a filtered report for the document type "XXXX" as the component name of the ABIE of the object class of the Document ABIE (e.g. OrderResponseSimple) in which all common library CCTS information not found to be used somewhere in the document type is not included

6.2.2 Model check report

Not included in the UBL distribution is a textual report the UBL TC utilizes as a "model check" that has analyzed the CCTS information reporting any violations of these Naming and Design Rules or any anomalies that may inadvertently interfere with the production of the validation artefacts. An example of such inadvertent interference is the use of a double-hyphen "--" in any description field. Such a sequence interferes with generating XML-based artefacts where some content might be, or later want to be, placed in an XML comment. Participants can be warned to revise field values so as not to include this or other sensitive character sequences.

7 User data validation artefacts

While the information bundles are described using CCTS principles, it is the document models that govern the structure of documents expressing in a machine-readable format the information described by the bundles. These NDR provide for expressing the model of an information bundle using validation artefacts that describe constraints on the structure and content of documents. Validation processes report violations of these model constraints found in a document.

There are two types of validation artefacts. W3C XSD schemas are used to validate the structure of elements and the structure of data content. OASIS CVA expressions and OASIS genericode files are used to validate the values of data content.

7.1 Rules for creating user data validation artefacts

These rules do not constrain the organization of the validation artefacts in any particular directory structure.

The CVA expressions typically are not used at runtime. More likely the CVA expressions and their associated genericode files would be processed or compiled into a runtime validation artefact. These rules do not constrain where this runtime artefact would be kept, but good practice suggests a location separate from the XSD schemas.

7.1.1 Namespaces

An important aspect of identifying and distinguishing the labels of information in information bundles is by using namespaces. Like-named constructs are distinguished by having different namespaces when using XML.

Note

Namespace abbreviations (also used here as namespace prefixes) in this document are not mandatory and have been selected solely for convenience and consistency. Implementations of these NDR and the documents governed by these NDR are welcome to use any namespace abbreviation or namespace prefix for any of the namespaces defined or referenced.

NAM01 Namespaces for information found in information bundles

A minimum set of namespaces is required for that information that is found in information bundles.

This set shall be as follows:

- one namespace for each document ABIE
  - these namespaces are not abbreviated in this document as they are not imported or included
- one namespace for all BBIE components
  - abbreviated in this document as "cbc" for "common basic components"
- one namespace for all ASBIE components (each component corresponding to a library ABIE)
  - abbreviated in this document as "cac" for "common aggregate components"
- one namespace for the extension collection and extension metadata elements

NAM01 Namespaces for information found in information bundles

- abbreviated in this document as "ext"

Each extension has a number of namespaces distinct from the document, library and other extensions.

NAM02 Namespaces for an extension

Each extension shall have a set of namespaces defined.

This set shall be as follows:

- one namespace for the apex ABIE of the extension
  - abbreviated in this document as "myext1"
- one namespace for all BBIE extension components that are not existing library components
  - abbreviated in this document as "mycbc1"
- one namespace for all ASBIE extension components (each corresponding to an extension ABIE) that are not existing library components
  - abbreviated in this document as "mycac1"

Note

The structure of an extension parallels that of a document with the distinct apex namespace, BBIE namespace and ASBIE namespace. Where possible the extension should reuse existing library components that have their own namespaces. This parallel approach makes it easier to consider incorporating extension elements in future versions of the library simply by changing the namespace.

Note

In these abbreviations "my" is used as in the first-person possessive pronoun, and "1" implies one of multiple extension namespaces

In addition to the namespaces for the labels of information bundles in documents, the description of the BBIE type information found in the bundles also is distinguished using namespaces. These namespaces are used only for labeling type information and not for labeling information items themselves and so are never found on XML elements in a document (thus they are never declared in an XML document governed by these NDR).

NAM03 Namespaces for BBIE data types

A minimum the following namespaces shall be defined for identifying type information for BBIEs.

This set shall be as follows:

- one namespace for all qualified data types
  - abbreviated in this document as "qdt"
- one namespace for all unqualified data types
  - abbreviated in this document as "udt"

NAM03 Namespaces for BBIE data types
• one namespace for CCTS Core Component Type definitions   • abbreviated in this document as "ccts-cct" **Note** These namespaces are used solely for describing the BBIE type information and are not ever needed in the XML document instances.

Schema annotations describing CCTS information require the use of a namespace.

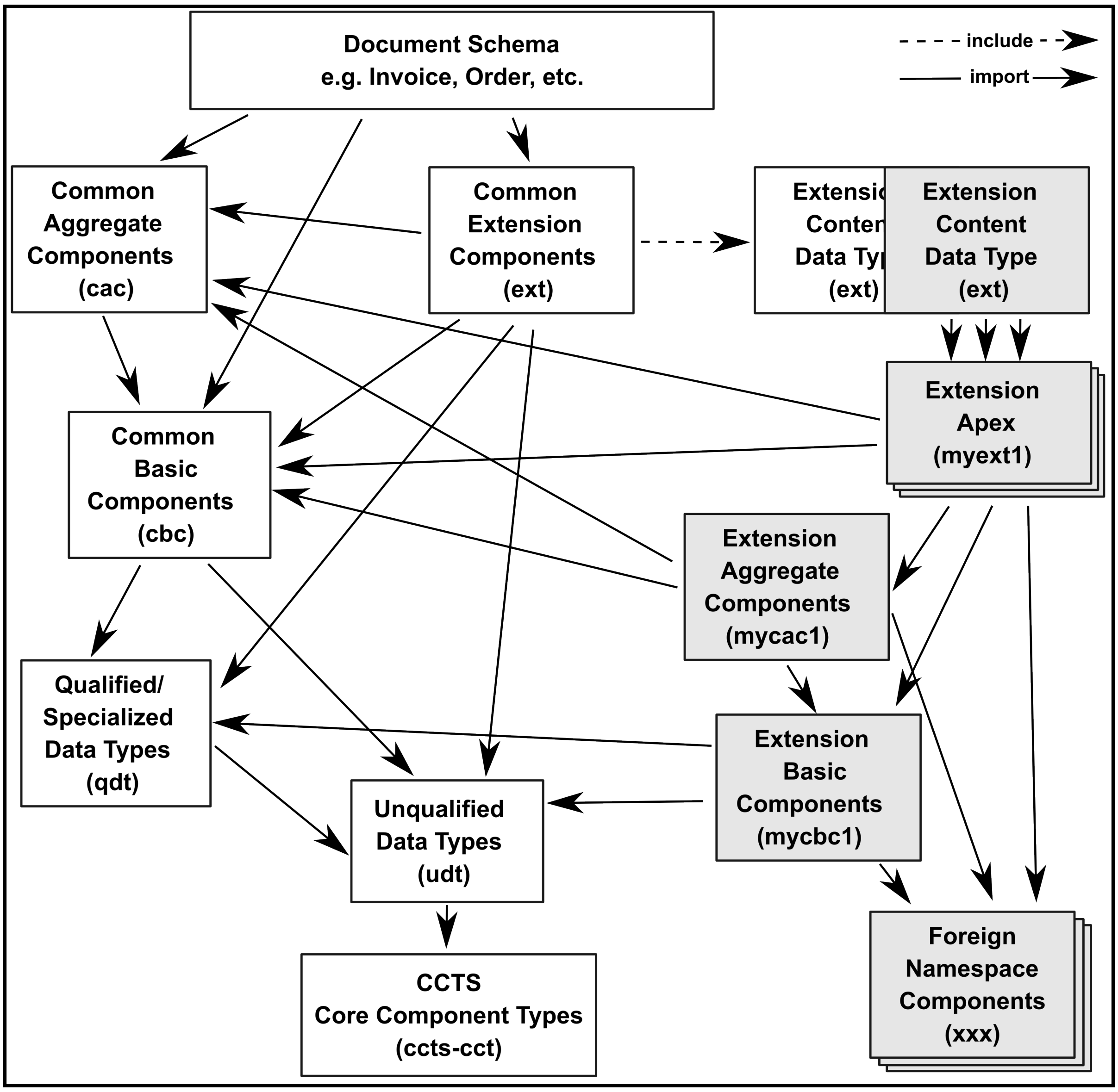
NAM04 Namespaces for schema annotations
When needed, a namespace for CCTS documentation found in schema annotations shall be defined as distinct from all other namespaces. This namespace is abbreviated in this document as "ccts".

An implementation of these NDR may choose to have additional namespaces for the management of type information.

7.1.2 XSD import/include tree

Figure 4, “Possible import/include hierarchy of XSD schema expressions” depicts the relationships between the W3C XSD fragments that are described in this section. There are no differences in the import/include hierarchy between the documented schemas and the runtime schemas. For reference purposes each box is a schema fragment and the parenthesized name tokens identify the namespace associated with the schema fragment.

Figure 4. Possible import/include hierarchy of XSD schema expressions


Document Schema
e.g. Invoice, Order, etc.
include
import
Common Aggregate Components (cac)
Common Extension Components (ext)
Extension Content Data Type (ext)
Extension Content Data Type (ext)
Common Basic Components (cbc)
Extension Apex (myext1)
Extension Aggregate Components (mycac1)
Qualified/ Specialized Data Types (qdt)
Extension Basic Components (mycbc1)
Unqualified Data Types (udt)
CCTS Core Component Types (ccts-cct)
Foreign Namespace Components (xxx)

In the diagram the unshaded boxes represent fragments created by the publishers of the schema. Once created there is no need for users of the schema to modify these fragments, and to ensure they are not inadvertently modified, these may be marked as read-only files in the file system.

The shaded boxes represent fragments modified and/or created by the users of the schema. The Extension Content Data Type is distinguished from other fragments in that it is initially created by the publishers of the schema and subsequently replaced by the users of the schema. This fragment defines which schemas are in play for the extension point of an extension item in the extension collection. The publishers of the schema

7.1.3 Schema fragment annotations

Provision is made in these NDR for annotating the schema fragments with model, housekeeping or maintenance information using XSD declaration annotations and XML comments. There are no constraints regarding which annotation information is added to the schema expressions. This information may be useful to the human reader or to tools exploiting the schema information in providing functionality to operators, but it does not impact on the interpretation of constraints imposed on XML documents being validated with the schema documents.

XSD annotation is typically defined for and with each of the many declarations in the file. Prudence suggests always providing a version of the published schema without XSD annotations so as not to burden runtime processing systems that provide no benefit having the information available. A runtime schema processor has no use of informational annotations and may incur unnecessary processing time ingesting and accommodating the information.

Separate from the concept of XSD annotations are simple XML comments that annotate schema fragments for identification or copyright purposes. Such comments are ignored by schema processors and do not burden their processing. The information in these comments is useful to users and, in some cases, may be required for intellectual property reasons imposed by the publisher. Good practice suggests that such information include module identification, module revision metadata and copyright declarations.

The built-in version annotation (the "version" attribute of the xsd:schema element) may be used to record the release version of the collection of schema fragments.

7.1.4 Library and document ABIE schema declarations

The schema fragment containing the definition of constraints for all common library ABIEs and each schema fragment containing the definition of constraints for a single document ABIE both have the same types of declarations.

DCL01 Element and type declarations
Every element and type declaration shall be global.

DCL02 Library ABIE schema contents
The one library ABIE schema shall include an element declaration for every library ABIE, every library ASBIE and every document ASBIE. It also includes a type declaration for every library ABIE.

DCL03 Document ABIE schema contents
Each document ABIE schema shall include a single element declaration, that being for the document ABIE. It also includes a single type declaration, that being for the document ABIE.

DCL04 Library and document ABIE schema element declarations

Each ABIE element shall be declared with the component name as the element name and the component name suffixed with "Type" as the type.

Example

```
<xsd:element name="WinningParty" type="WinningPartyType"/>
```

DCL05 Library and document ASBIE schema element declarations

Each ASBIE element shall be declared with the component name as the element name and the component name of the ABIE of the associated object class suffixed with "Type" as the type.

Example

```
<xsd:element name="AgentParty" type="PartyType"/>
```

DCL06 Library and document ABIE schema type declarations

The library ABIE schema shall include a complex type declaration for every library ABIE. The document ABIE schema shall include a complex type declaration for the document ABIE.

Each complex type name shall be declared with the component name of the ABIE suffixed with "Type" as the type.

Example

```
<xsd:complexType name="WinningPartyType">
 <xsd:sequence>
  <xsd:element ref="cbc:Rank" minOccurs="0" maxOccurs="1"/>
  <xsd:element ref="cac:Party" minOccurs="1" maxOccurs="1"/>
 </xsd:sequence>
</xsd:complexType>
```

DCL07 ABIE schema type content declaration order

The members of an ABIE shall be ordered as a sequence with all BBIE element declarations first, followed by all ASBIE declarations.

Example

```
<xsd:complexType name="CatalogueRequestLineType">
 <xsd:sequence>
  <xsd:element ref="cbc:ID" minOccurs="1" maxOccurs="1"/>
  <xsd:element ref="cbc:ContractSubdivision"
               minOccurs="0" maxOccurs="1"/>
  <xsd:element ref="cbc:Note" minOccurs="0"
               maxOccurs="unbounded"/>
  <xsd:element ref="cac:LineValidityPeriod"
               minOccurs="0" maxOccurs="1"/>
  <xsd:element ref="cac:RequiredItemLocationQuantity"
               minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element ref="cac:Item" minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

DCL08 Document ABIE schema type extension declaration

The members of a document ABIE that is supporting extensions shall be preceded by a reference to the extension collection element declaration.

Example

```
<xsd:complexType name="ApplicationResponseType">
  <xsd:sequence>
     <xsd:element ref="ext:UBLExtensions"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:UBLVersionID"
                      minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:CustomizationID"
                      minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:ProfileID"
```

DCL08 Document ABIE schema type extension declaration

```
                      minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:ProfileExecutionID"
                      minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:ID"
                     minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="cbc:UUID"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:IssueDate"
                     minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="cbc:IssueTime"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:ResponseDate"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:ResponseTime"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:Note"
                     minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="cbc:VersionID"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cac:Signature"
                     minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="cac:SenderParty"
                      minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="cac:ReceiverParty"
                     minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="cac:DocumentResponse"
                     minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Note

The rationale for positioning extension information at the very start of the XML instance is to allow processing applications acting sequentially on the document to consume and cache all non-modeled extension information in preparation for consuming any of the modeled document information. Should extension information follow modeled information, the sequential processing of the modeled information would have passed before recognizing the need to associate extension information. In essence, such a sequential processing application would have to cache the entire document, thus losing the benefit of sequential processing.

DCL09 ABIE schema type content declaration style

The members of an ABIE shall be declared using references to element declarations.

Example

```
<xsd:complexType name="ItemIdentificationType">
  <xsd:sequence>
     <xsd:element ref="cbc:ID" minOccurs="1" maxOccurs="1"/>
     <xsd:element ref="cbc:ExtendedID"
                     minOccurs="0" maxOccurs="1"/>
     <xsd:element ref="cbc:BarcodeSymbologyID"
                      minOccurs="0" maxOccurs="1"/>
     <xsd:element ef="cac:PhysicalAttribute"
```

DCL09 ABIE schema type content declaration style

```
                     minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="cac:MeasurementDimension"
                     minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="cac:IssuerParty"
                     minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

There are no constraints on the order of the declarations in the schema expression.

Note

This lack of an order constraint may seem in conflict with the library constraint of alphabetical order of model components. Whereas the model components are organized for the benefit of the human reader, the order of the schema component declarations does not influence the behaviour of a schema processor. Choosing alphabetical order would be a convenience for the unexpected human reader of the schema expressions.

7.1.5 Library BBIE schema declarations

DCL10 Library BBIE schema

The library BBIE schema shall have element declarations for every BBIE referenced in all document ABIEs and all library ABIEs.

DCL11 Library BBIE schema

The library BBIE schema shall have a complex type declaration for every BBIE referenced in all document ABIEs and all library ABIEs.

DCL12 Library BBIE schema element declarations

Each BBIE element shall be declared with the component name as the element name and the concatenation of the component name and "Type" as the type.

Example

```
<xsd:element name="SourceCurrencyCode"
             type="SourceCurrencyCodeType"/>
```

DCL13 Library BBIE schema element declarations

Each BBIE element type shall be declared as simple content extended from a base of either a qualified data type or an unqualified data type without the addition of any additional attributes.

Example

```
<xsd:complexType name="SourceCurrencyCodeType">
   <xsd:simpleContent>
      <xsd:extension base="udt:CodeType"/>
   </xsd:simpleContent>
</xsd:complexType>
```

7.1.6 Extension collection schema declarations

The validation artefacts may contain an extension collection in order to provide for the inclusion of extra-model information in a document. Such information includes content standardized by other organizations (e.g. signature information), and augmentations of the document models.

The extension collection contains one or more extension items. Each extension item has extension metadata elements may reuse existing ABIEs or BBIEs from the common library or may contain specific metadata unrelated to concepts found in the common library.

EXT01 Extension collection cardinality
A document may be allowed to contain at most one extension collection, and if it is, that extension collection shall be optional and not repeated. **Note** The rationale for the item being optional is that any given instance of the document may not have any extension information to augment the modeled business objects.

EXT02 Extension collection location
The document's extension collection shall be realized in the validation artefacts as the first child of the document ABIE validation construct, before that of any BIE modeled for the document ABIE. **Note** The rationale is to support serial processing of the information. All extension information is encountered, and cached if necessary by a program, before encountering any of the modeled document information. This ensures the program has all information about a modeled business object when encountering that object in the serial stream of content, regardless of the presence or absence of extension information.

EXT03 Extension collection content
The document's extension collection shall require one or more extension items as its content. **Note** The rationale is that different users of a document may have different extension items added to the content. Also, different extensions may be thematically distinguished (e.g. the digital signature extension is semantically separate from an extension augmenting invoice line content).

EXT04 Extension item content
The extension item shall have a collection of metadata information regarding the given instance of the item, plus a mandatory single element being the extension point under which extension information is added to the document.

EXT05 Extension point
The extension point of each extension item shall require a single element in some namespace other than the extension namespace. There shall be no other constraints imposed on the extension point element.

EXT06 Extension collection and item schema fragment and declarations

The extension collection schema fragment shall include the declarations of the extension collection element, the extension item element, and the extension item metadata elements and any required type information for locally-defined elements.

The extension collection schema fragment shall include the declaration of the mandatory extension point element, but not the type information for the extension point element.

Note

The rationale for not including the type information for the extension point element is that the type is subject to change by a user of the schemas, while the extension collection, the extension item and extension item metadata element and type information is not. This separation allows the extension collection and item schema fragment to be deployed as read-only, while the extension point data type schema fragment can be deployed as writable in order to be defined by users.

EXT07 Extension point data type schema fragment and declaration

The extension point schema fragment shall contain only a single complex type declaration being a sequence of exactly one element in a namespace other than the extension namespace to be processed with "lax validation".

Example

```
<xsd:complexType name="ExtensionContentType">
  <xsd:sequence>
    <xsd:any namespace="##other" processContents="lax"
             minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
```

Note

The rationale for the lax validation is to allow for the extension point to contain, without error, any element for which there are no constraints in any schema fragment imported or included in the validation step.

EXT08 Extension point data type schema fragment import directives

The extension point schema fragment may contain as many import directives for extension point schemas as desired. There is no order to the directives.

7.1.7 Qualified Data Types schema declarations

Any required qualified data types are declared in the Qualified Data Types module.

QDT01 Qualified Data Type declaration name

Every qualified data type shall be declared using the name of the data type qualifier followed by the unqualified data type name.

QDT02 Qualified Data Type declaration basis

Every qualified data type shall be based on an unqualified data type, imposing whatever qualifications are required to be expressed using XSD schema semantics.

QDT02 Qualified Data Type declaration basis
The resulting declaration shall be such that every possible instance of the declared complex type is also an instance of the base type.   **Note**   This constraint prevents additions of anything that is not part of the base type, such as the introduction of any new attributes or sub-elements, or any less-constrained element or attribute values.

7.1.8 Unqualified Data Types schema declarations

All unqualified data types are declared in the Unqualified Data Types module.

UDT01 Unqualified Data Type declaration suite
An unqualified data type shall be declared for every one of and only the 20 CCTS Primary Representation Terms and Secondary Representation Terms defined in Core Component Technical Specification Section 8-3 "Permissible Representation Terms".

UDT02 Unqualified Data Type declaration base
Every unqualified data type shall be based on one of the 10 CCTS Core Component Type schema data types defined in Core Component Technical Specification Section 8-2 "Approved Core Component Type Content and Supplementary Components".   This may be accomplished by either restricting a declaration imported from the Core Component Types schema or by wholly replacing the corresponding Core Component Types schema declaration.   The resulting declaration shall be such that every possible instance of the declared complex type is also an instance of the base type.   **Note**   The rational for having UDT declarations is to impose some XSD syntax semantics on top of the more general decimal and string lexical syntax defined in the CCTS specification of Core Component Types. For example, the CCTS Core Component Type for date and time is a simple string without constraint. Such lax structuring of the field value does not serve users in that no particular format is obligated. The UDT can impose, for example, the xsd:dateTime lexical syntax on all date and time values, overriding the CCTS definition.   **Note**   This rule does allow an optional supplementary component defined in the CCTS Core Component Type not to be available in the associated unqualified data type. For example, if the UDT implements a built-in XSD data type for the component then there is no use of a format supplementary component and associated attribute and so the format attribute declaration can be omitted and, thereby, be unavailable for use for that data type.   **Example 1**   In this example the unqualified data type uses the base type without change:

UDT02 Unqualified Data Type declaration base
(continued)

```
<xsd:complexType name="NumericType">
  <xsd:simpleContent>
    <xsd:extension base="ccts-cct:NumericType"/>
  </xsd:simpleContent>
</xsd:complexType>
```

Example 2

In this example the unqualified data type redeclares the base type's optional attribute as mandatory:

```
<xsd:complexType name="MeasureType">
  <xsd:simpleContent>
    <xsd:restriction base="ccts-cct:MeasureType">
      <xsd:attribute name="unitCode"
                     type="xsd:normalizedString"
                     use="required"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>
```

Example 3

In this example the unqualified data type replaces the base type with no attrib-utes and with an XSD built-in type for content:

```
<xsd:complexType name="DateTimeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:dateTime"/>
  </xsd:simpleContent>
</xsd:complexType>
```

7.1.9 CCTS Core Component Types schema

All primitive data types correspond to the 10 CCTS Primary Representation Terms defined in Core Component Technical Specification Section 8-3 "Permissible Representation Terms".

CCT01 CCTS Core Component Type schema
The core component type schema of primitive data types for primary representation terms on which the unqualified data types are based shall be an unmodified copy of a schema fragment as published by UN/CEFACT.

7.1.10 Data type qualifications

Data type qualifications that are not expressed as qualified data types using XSD schema semantics are expressed using the OASIS Context Value Association [CVA] XML vocabulary. The CVA file provides for users to associate value evaluation expressions and OASIS genericode code lists with different CVA contexts.

DTQ01 Data type qualifications for element content
A CVA context shall be created for every BBIE with a non-empty value in the CCTS information for the qualified data type.

DTQ02 Data type qualifications for attribute content
A CVA context shall be created for every CCTS supplementary component to be validated.

7.2 UBL creation of user data validation artefacts (Non-Normative)

The UBL distribution organizes the documented XSD schemas (with BIE XSD annotations) under the `xsd/` subdirectory and the runtime XSD schemas (without BIE XSD annotations) under the `xsdrt/` subdirectory:

> http://docs.oasis-open.org/ubl/os-UBL-2.1/xsd/
>
> http://docs.oasis-open.org/ubl/os-UBL-2.1/xsdrt/

The CVA file and formatted report are distributed in the `cva/` subdirectory:

> http://docs.oasis-open.org/ubl/os-UBL-2.1/cva/

The genericode files referenced by the CVA file are distributed in the `gc/` subdirectories of both the UBL 2.1 distribution and the UBL 2.0 distribution:

> http://docs.oasis-open.org/ubl/os-UBL-2.1/cl/gc/
>
> http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/

The UBL CVA and genericode files are compiled into an XSLT stylesheet expression for runtime testing against XML instances that have been validated using the UBL XSD schemas. This stylesheet is named `UBL-DefaultDTQ-2.1.xsl` and is distributed in the `val/` subdirectory:

> http://docs.oasis-open.org/ubl/os-UBL-2.1/val/

7.2.1 UBL namespaces

The following namespace URI strings are associated with the indicated documentary namespace prefixes:

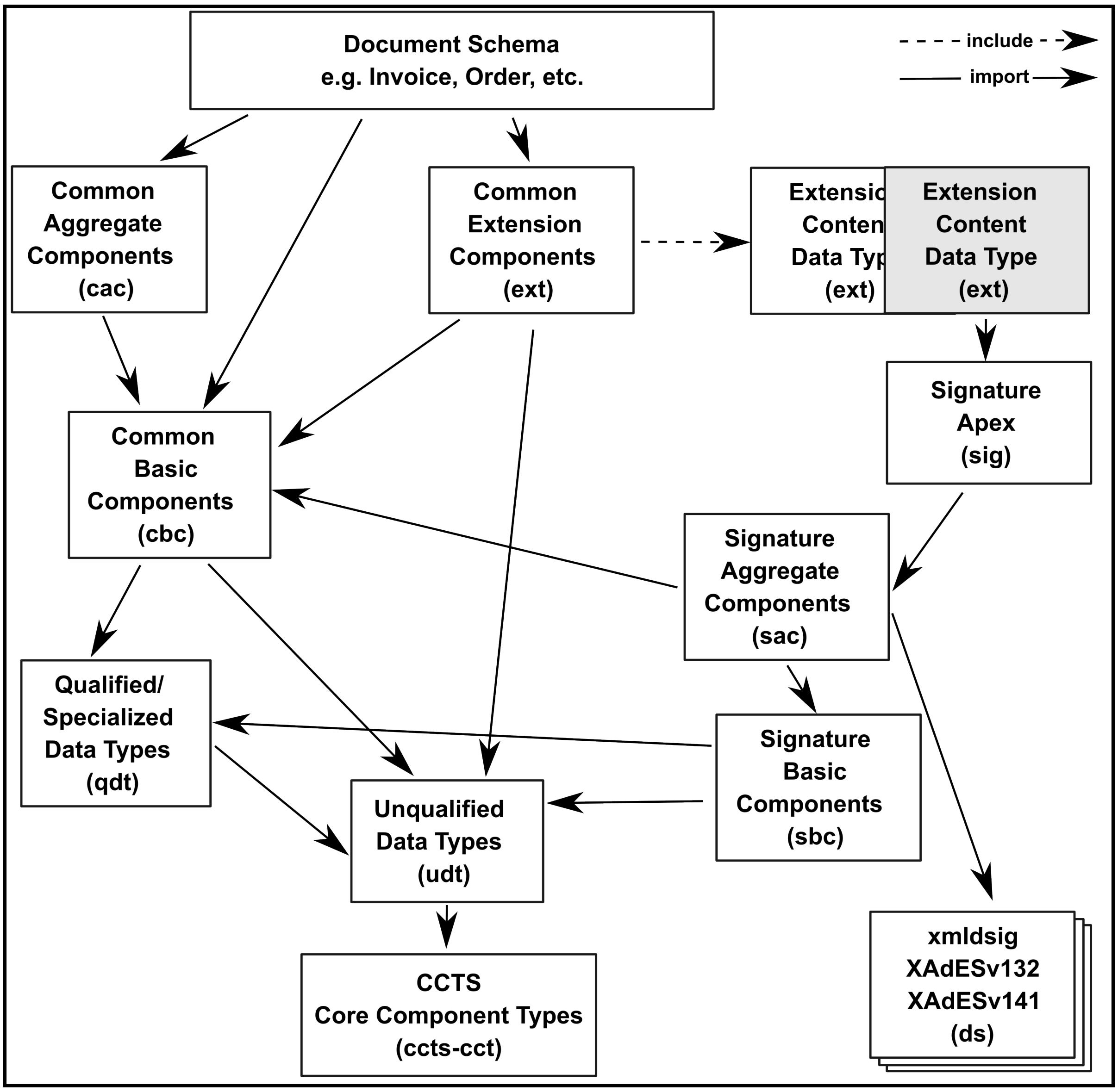
Table 2. UBL Namespaces

(no prefix)	Document ABIE namespaces use the component name of the ABIE XXXXXXXX as follows: `urn:oasis:names:specification:ubl:schema:xsd:XXXXXXXX-2`
cbc	`urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2`
cac	`urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2`
ext	`urn:oasis:names:specification:ubl:schema:xsd:CommonExtensionComponents-2`
sig	`urn:oasis:names:specification:ubl:schema:xsd:CommonSignatureComponents-2`
sac	`urn:oasis:names:specification:ubl:schema:xsd:SignatureAggregateComponents-2`
sbc	`urn:oasis:names:specification:ubl:schema:xsd:SignatureBasicComponents-2`
qdt	`urn:oasis:names:specification:ubl:schema:xsd:QualifiedDataTypes-2`
udt	`urn:oasis:names:specification:ubl:schema:xsd:UnqualifiedDataTypes-2`
ccts-cct	`urn:un:unece:uncefact:data:specification:CoreComponentTypeSchemaModule:2`
ccts	`urn:un:unece:uncefact:documentation:2`

7.2.2 UBL XSD import/include tree

Figure 5, "UBL import/include hierarchy of XSD schema expressions" depicts the UBL implementation of the import and include directives shown in Figure 4, "Possible import/include hierarchy of XSD schema expressions".

Figure 5. UBL import/include hierarchy of XSD schema expressions


Document Schema
e.g. Invoice, Order, etc.
include
import
Common Aggregate Components (cac)
Common Extension Components (ext)
Extension Content Data Type (ext)
Extension Content Data Type (ext)
Signature Apex (sig)
Common Basic Components (cbc)
Signature Aggregate Components (sac)
Qualified/ Specialized Data Types (qdt)
Signature Basic Components (sbc)
Unqualified Data Types (udt)
xmldsig
XAdESv132
XAdESv141
(ds)
CCTS
Core Component Types
(ccts-cct)

7.2.3 UBL common schema fragment annotations

The documented XSD schemas include CCTS information in annotations of the Library and Document ABIE complex types (not of the element declarations). Only those values found in the columns referenced in Section 4.1.4, "BIE definition conventions" are included. An example of each of an ABIE, BBIE and ASBIE is as follows:

```
<xsd:complexType name="WinningPartyType">
  <xsd:annotation>
     <xsd:documentation>
        <ccts:Component>
           <ccts:ComponentType>ABIE</ccts:ComponentType>
           <ccts:DictionaryEntryName>Winning Party.
           Details</ccts:DictionaryEntryName>
           <ccts:Definition>A party that is identified as the awarded by
           a tender result.</ccts:Definition>
           <ccts:ObjectClass>Winning Party</ccts:ObjectClass>
        </ccts:Component>
     </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
     <xsd:element ref="cbc:Rank" minOccurs="0" maxOccurs="1">
        <xsd:annotation>
           <xsd:documentation>
              <ccts:Component>
                 <ccts:ComponentType>BBIE</ccts:ComponentType>
                 <ccts:DictionaryEntryName>Winning Party. Rank.
                 Text</ccts:DictionaryEntryName>
                 <ccts:Definition>Indicates the rank obtained in the
                 award.</ccts:Definition>
                 <ccts:Cardinality>0..1</ccts:Cardinality>
                 <ccts:ObjectClass>Winning Party</ccts:ObjectClass>
                 <ccts:PropertyTerm>Rank</ccts:PropertyTerm>
                 <ccts:RepresentationTerm>Text</ccts:RepresentationTerm>
                 <ccts:DataType>Text. Type</ccts:DataType>
              </ccts:Component>
           </xsd:documentation>
        </xsd:annotation>
     </xsd:element>
     <xsd:element ref="cac:Party" minOccurs="1" maxOccurs="1">
        <xsd:annotation>
           <xsd:documentation>
              <ccts:Component>
                 <ccts:ComponentType>ASBIE</ccts:ComponentType>
                 <ccts:DictionaryEntryName>Winning Party.
                 Party</ccts:DictionaryEntryName>
                 <ccts:Definition>Information about an organization,
                 sub-organization, or individual fulfilling a role in a
                 business process.</ccts:Definition>
                 <ccts:Cardinality>1</ccts:Cardinality>
                 <ccts:ObjectClass>Winning Party</ccts:ObjectClass>
                 <ccts:PropertyTerm>Party</ccts:PropertyTerm>
                 <ccts:AssociatedObjectClass>Party</ccts:Associated
ObjectClass>
                 <ccts:RepresentationTerm>Party</ccts:RepresentationTerm>
              </ccts:Component>
           </xsd:documentation>
        </xsd:annotation>
     </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Every schema fragment generated for UBL 2.1 includes an XML comment at both the beginning of the document and the end of the document. These comments are present in both the documented schemas and the runtime schemas.

The comment at the beginning of every schema fragment includes identification information and revision metadata, as in this example from the common aggregate components fragment:

```
<!--
  Library:           OASIS Universal Business Language (UBL) 2.1 OS
                     http://docs.oasis-open.org/ubl/os-UBL-2.1/
  Release Date:      04 November 2013
  Module:            xsd/common/UBL-CommonAggregateComponents-2.1.xsd
  Generated on:      2013-10-31 17:17z
  Copyright (c) OASIS Open 2013. All Rights Reserved.
-->
```

The comment at the end of the document is a detailed copyright notice as follows:

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

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

  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 OASIS,
  except as needed for the purpose of developing OASIS
  specifications, in which case the procedures for copyrights defined
  in the OASIS Intellectual Property Rights document must be
  followed, or as required to translate it into languages other than
  English.

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

  This document and the information contained herein is provided on
  an "AS IS" basis and OASIS 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.
-->
```

Throughout the fragments are short XML comments delineating sections with declarations for different purposes.

The built-in version annotation of the schema document element is used to reflect the version of the UBL release. For example, this is used for every schema fragment in the UBL 2.1 release:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="2.1" ...
```

7.2.4 UBL library and document ABIE schema-declarations

The UBL common ABIE library schema fragment is named UBL-CommonAggregateComponents-2.1.xsd in UBL 2.1.

Each UBL document ABIE schema fragment is named UBL-XXXXX-2.1.xsd in UBL 2.1, where XXXXX is the component name of the document ABIE.

There is only one addition in the UBL implementation of declarations as in Section 7.2.4, “UBL library and document ABIE schema-declarations”. The element declaration in a document ABIE includes an annotation as follows:

```
<xsd:annotation>
   <xsd:documentation>This element MUST be conveyed as the root element
in any instance document based on this Schema expression</xsd:documentation>
</xsd:annotation>
```

7.2.5 UBL library BBIE declarations

The UBL common BBIE library schema fragment is named UBL-CommonBasicComponents-2.1.xsd in UBL 2.1.

There are no differences in the UBL implementation of declarations as in Section 7.1.5, “Library BBIE schema declarations”.

7.2.6 UBL extension collection schema declarations

The UBL common extension library schema fragment is named UBL-CommonExtensionComponents-2.1.xsd in UBL 2.1.

The UBL extension collection element is named "UBLExtensions" and its type is named "UBLExtensionsType". It is comprised of one or more UBL extension elements.

The UBL extension element is named "UBLExtension" and its type is named " UBLExtensionType". It is comprised of the following elements, in order, each with a type named by the element name suffixed with "Type", each with a cardinality of 0..1, and those declared in the module are declared with the given unqualified data type:

- cbc:ID (from the common library)
  - An identifier for the Extension assigned by the creator of the extension.
- cbc:Name (from the common library)
  - A name for the Extension assigned by the creator of the extension.
- ExtensionAgencyID (of unqualified data type "Identifier. Type")
  - An agency that maintains one or more Extensions.
- ExtensionAgencyName (of unqualified data type "Text. Type")

    - The name of the agency that maintains the Extension.
- ExtensionVersionID (of unqualified data type "Identifier. Type")
    - The version of the Extension.
- ExtensionAgencyURI (of unqualified data type "Identifier. Type")
    - A URI for the Agency that maintains the Extension.
- ExtensionURI (of unqualified data type "Identifier. Type")
    - A URI for the Extension.
- ExtensionReasonCode (of unqualified data type "Code. Type")
    - A code for reason the Extension is being included.
- ExtensionReason (of unqualified data type "Text. Type")
    - A description of the reason for the Extension.
- ExtensionContent (this type is declared in an included extension content data type schema fragment)
    - The definition of the extension content.

The UBL implementation of the extension content data type schema fragment imposes "lax" processing on a single, mandatory element of type xsd:any. The schema fragment imports the UBL common digital signature extension in order to impose that extension's schema constraints when the signature extension element is used.

The UBL common digital signature extension schema fragment is named UBL-CommonSignatureComponents-2.1.xsd in UBL 2.1.

7.2.7 UBL Qualified Data Type schema declarations

The UBL qualified data type schema fragment is named UBL-QualifiedDataTypes-2.1.xsd in UBL 2.1.

The UBL implementation of qualified data types is empty. There are no qualified data types. Data type qualifications are validated solely through the expression of code list constraints using CVA expressions and genericode code list expressions.

7.2.8 UBL Unqualified Data Type schema declarations

The UBL unqualified data type schema fragment is named UBL-UnqualifiedDataTypes-2.1.xsd in UBL 2.1.

The following modifications of the CCTS Core Component Types are implemented for the 20 CCTS Primary Representation Terms and Secondary Representation Terms defined in Core Component Technical Specification Section 8-3 "Permissible Representation Terms". The modifications are limited to changing the cardinality of an attribute from optional to required and imposing a particular lexical format in place of allowing a format identifier.

Table 3. Core Component Type constraints in UBL

Amount Type	Amount. Currency. Identifier required as `@currencyID`
Binary Object Type	Binary Object. Mime. Code required as `@mimeCode`
Graphic Type	Graphic. Mime. Code required as `@mimeCode`
Picture Type	Picture. Mime. Code required as `@mimeCode`
Sound Type	Sound. Mime. Code required as `@mimeCode`
Video Type	Video. Mime. Code required as `@mimeCode`
Code Type	*(unchanged)*
Date Time Type	Date Time. Format. Text omitted; Date Time. Type based on `xsd:dateTime`
Date Type	Date Time. Format. Text omitted; Date Time. Type based on `xsd:date`
Time Type	Date Time. Format. Text omitted; Date Time. Type based on `xsd:time`
Identifier Type	*(unchanged)*
Indicator Type	Indicator. Format. Text omitted; Indicator. Type based on `xsd:dateTime`
Measure Type	Measure. Unit. Code required as `@unitCode`
Numeric Type	*(unchanged)*
Value Type	*(unchanged)*
Percent Type	*(unchanged)*
Rate Type	*(unchanged)*
Quantity Type	*(unchanged)*
Text Type	*(unchanged)*
Name Type	*(unchanged)*

7.2.9 UBL CCTS Core Component Types schema

The UBL CCTS Core Component Types schema fragment is named CCTS_CCT_SchemaModule-2.1.xsd in UBL 2.1.

The CCTS Core Component Types schema fragment version 1.1, copyrighted in 2006 by UN/CEFACT, inspired the creation of the unqualified data types in UBL 2.0 and was included in the 2006 UBL distribution. These data types include content and supplementary components declared as normalized strings.

For backward compatibility, UBL 2.1 bases the unqualified data types on the same version 1.1 schema fragment in its schema import hierarchy and not the version 2.0 schema fragment published in 2007.

Important note regarding compatibility

Version 2.0 of the XSD for Core Component Types is not backward compatible with version 1.1 of the XSD for Core Component Types. An example of being unable to replace version 1.1 with version 2.0 is the declaration of content and components as XSD tokens. Tokens are more restricted than the normalized strings used in version 1.1, thus a value expressed using 1.1 may not validate using 2.0 schemas. Any schema import tree based on version 2.0 of the XSD for Core Component Types will not be backward compatible with the same import tree based on version 1.1.

The version 1.1 schema begins with the following metadata:

```
<!-- ======================================================================= -->
<!-- ===== CCTS Core Component Type Schema Module ===== -->
<!-- ======================================================================= -->
<!--
```

```
   Module of Core Component Type
   Agency: UN/CEFACT
   VersionID: 1.1
   Last change: 14 January 2005



   Copyright (C) UN/CEFACT (2006). All Rights Reserved.
```

Important note regarding access

This module is periodically revised by UN/CEFACT and archive copies of previous versions appear not to be available on the UN/CEFACT web site. A copy of version 1.1 of the XSD for Core Component Types cannot be found on the UN/CEFACT web site but is found in both of the UBL distributions at:

http://docs.oasis-open.org/ubl/os-UBL-2.0/xsd/common/CCTS_CCT_SchemaModule-2.0.xsd

http://docs.oasis-open.org/ubl/os-UBL-2.1/xsd/common/CCTS_CCT_SchemaModule-2.1.xsd

7.2.10 UBL data type qualifications

The data type qualifications in UBL 2.1 reference code lists whose name matches a name found in the Data Type Qualification information item of all BBIEs. These code lists are found in the following directories:

- UBL 2.0 supplementary components
  - http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/cefact/
- UBL 2.0 qualifications
  - http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/default/
- UBL 2.1 qualifications
  - http://docs.oasis-open.org/ubl/os-UBL-2.1/cl/gc/default/

For completeness, the Binary Object. Mime. Code and Measure. Unit. Code lists, using the names "BinaryObjectMime" and "UnitOfMeasure" respectively, also are included from these directories:

- http://docs.oasis-open.org/ubl/os-UBL-2.0/cl/gc/cefact/
- http://docs.oasis-open.org/ubl/os-UBL-2.1/cl/gc/default/

Instance metadata is described for the supplementary components as follows:

```
<InstanceMetadataSets>
<Annotation>
  <Description>
    <x:p>
      A supplementary component is a piece of metadata describing a
      property of the list of values in which a particular value is
      found.  For example, each code list of currency values has a
      version and when a currency value is specified in a UBL document
      it may be important to specify which version of the currency
      code list that value is found.  By knowing the list from which a
      value is found, the semantics representated by that value are
      unambiguous.
    </x:p>
    <x:p>
      These sets of supplementary components describe where instance-level
      metadata is found that contains values of list-level metadata from
      genericode files.  Each set is identified in a context using that
      set's unique <x:samp>xml:id=</x:samp> attribute.
    </x:p>
    <x:p>
      Each component pairs an XPath address of the supplementary component
      (specified in <x:samp>address=</x:samp>) relative to the item with
      the code list value in the UBL document, with the XPath address of
      the corresponding list-level metadata item in the genericode file
      (specified in <x:samp>identification=</x:samp>) relative to the
      <x:samp>&lt;Identification></x:samp> element in the code
      list file.
    </x:p>
    <x:p>
      All supplementary components in UBL are defined by the UN/CEFACT Core
      Component Technical Specification (CCTS) Version 2.01 dated
      15 November 2003
      <x:a href="http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.pdf">
    <x:samp>http://www.unece.org/cefact/ebxml/CCTS_V2-01_Final.
              pdf</x:samp>.</x:a>
      Table 8-2 "Approved Core Component Type Content
      and Supplementary Components" lists the various supplementary
      components available.  The UN/CEFACT
      UnqualifiedDataTypeSchemaModule-2.0.xsd schema fragment specifies
      the names of the attributes for supplementary components.
    </x:p>
  </Description>
</Annotation>
<InstanceMetadataSet xml:id="cctsV2.01-amount">
  <Annotation>
    <Description>
      <x:p>
        The <x:samp>Amount. Type</x:samp> supplementary component.
      </x:p>
    </Description>
  </Annotation>
  <InstanceMetadata address="../@currencyCodeListVersionID"
                    identification="Version"/>
</InstanceMetadataSet>
<InstanceMetadataSet xml:id="cctsV2.01-binary">
  <Annotation>
    <Description>
      <x:p>
        There are no supplementary components for the
        <x:samp>Binary Object. Type</x:samp>.
      </x:p>
    </Description>
  </Annotation>
</InstanceMetadataSet>
<InstanceMetadataSet xml:id="cctsV2.01-code">
  <Annotation>
    <Description>
      <x:p>
        The <x:samp>Code. Type</x:samp> supplementary components.
      </x:p>
    </Description>
  </Annotation>
  <InstanceMetadata address="@listName"
                    identification="LongName[not(@Identifier='listID')]"/>
  <InstanceMetadata address="@listID"
                    identification="LongName[@Identifier='listID']"/>
  <InstanceMetadata address="@listVersionID"
                    identification="Version"/>
  <InstanceMetadata address="@listSchemeURI"
                    identification="CanonicalUri"/>
  <InstanceMetadata address="@listURI"
                    identification="LocationUri"/>
  <InstanceMetadata address="@listAgencyName"
                    identification="Agency/LongName"/>
  <InstanceMetadata address="@listAgencyID"
                    identification="Agency/Identifier"/>
</InstanceMetadataSet>
<InstanceMetadataSet xml:id="cctsV2.01-identifier">
  <Annotation>
    <Description>
      <x:p>
        The <x:samp>Identifier. Type</x:samp> supplementary components.
      </x:p>
    </Description>
  </Annotation>
  <InstanceMetadata address="@schemeName"
                    identification="LongName[not(@Identifier='listID')]"/>
  <InstanceMetadata address="@schemeID"
                    identification="LongName[@Identifier='listID']"/>
  <InstanceMetadata address="@schemeVersionID"
                    identification="Version"/>
  <InstanceMetadata address="@schemeURI"
                    identification="CanonicalUri"/>
  <InstanceMetadata address="@schemeDataURI"
                    identification="LocationUri"/>
  <InstanceMetadata address="@schemeAgencyName"
                    identification="Agency/LongName"/>
  <InstanceMetadata address="@schemeAgencyID"
                    identification="Agency/Identifier"/>
</InstanceMetadataSet>
<InstanceMetadataSet xml:id="cctsV2.01-measure">
  <Annotation>
    <Description>
      <x:p>
        The <x:samp>Measure. Type</x:samp> supplementary component.
      </x:p>
    </Description>
  </Annotation>
  <InstanceMetadata address="../@unitCodeListVersionID"
                    identification="Version"/>
</InstanceMetadataSet>
<InstanceMetadataSet xml:id="cctsV2.01-quantity">
  <Annotation>
    <Description>
      <x:p>
        The <x:samp>Quantity. Type</x:samp> supplementary components.
      </x:p>
    </Description>
  </Annotation>
  <InstanceMetadata address="../@unitCodeListID"
                    identification="LongName[@Identifier='listID']"/>
  <InstanceMetadata address="../@unitCodeListAgencyName"
                    identification="Agency/LongName"/>
  <InstanceMetadata address="../@unitCodeListAgencyID"
                    identification="Agency/Identifier"/>
</InstanceMetadataSet>
</InstanceMetadataSets>
```

The following contexts are described for attributes:

- every `currencyID` attribute is a context for the currency code lists
- every `mimeCode` attribute is a context for the binary object mime code lists
- every `unitCode` attribute of those BBIEs with a Data Type value of "Measure" is a context for the unit of measure code lists
- every `unitCode` attribute of those BBIEs with a Data Type value of "Quantity" is a context for the unit of measure code lists

The following contexts are described for elements:

- every element of those BBIEs with a Qualified Data Type value is a context for a code list with the same name as the value

8 Conformance

A set of information bundle definitions and their associated user data validation artefacts conforming to these naming and design rules does not violate any rule or requirement expressed in normative sections of this specification where the word *shall* is used in the context for which the user is utilizing these rules.

Note

The different contexts referenced here relate to the different examples described in Section 1, “Introduction” of how users may wish to utilize this specification.

Note

The non-normative sections of this specification that document the tools and techniques the UBL Technical Committee used to produce the UBL 2.1 validation artefacts do not govern conformance. There is no obligation for users of this specification to use the same tools and techniques. There is only the obligation to produce information bundle definitions and associated user data validation artefacts in such a way as not to violate any normative rule for the context the user wishes to utilize.

Appendix A Release notes (Non-Normative)

A.1 Availability

Online and downloadable versions of this release are available from the locations specified at the top of this document.

A.2 Status of this release

Release of this package to the public marks the beginning of its first public review. The UBL Technical Committee actively solicits input from the user community regarding this release. See Status at the beginning of this document for procedures to be used in submitting comments to the Committee. Note that in accordance with OASIS policies regarding intellectual property, the UBL TC *cannot* accept input from persons outside the UBL TC (including OASIS members) unless it is submitted via the comment list.

THIS RELEASE IS SUBJECT TO CHANGE. IT IS PROVIDED FOR TESTING PURPOSES ONLY AND SHOULD NOT BE USED FOR PRODUCTION SYSTEMS.

A.3 Package structure

This Committee Specification Draft 01 / Public Review Draft 01 is published as a zip archive in the http://docs.oasis-open.org/ubl/UBL-NDR/v3.0/csprd01/ directory. Unzipping this archive creates a directory tree containing a number of files and subdirectories. Note that while the two XML files comprise the revisable version of this specification, this revisable XML may not be directly viewable in all currently available web browsers.

The base directory has the following files:

`UBL-NDR-v3.0-csprd01.xml`
The revisable form of the document.

`UBL-NDR-v3.0-csprd01-summary.xml`
A distillation of the rules, comprising only the rule title and first paragraph. This XML document is incorporated in the revisable form of the document by way of an entity reference. During the publishing process this file is first distilled from the revisable form and then subsequently incorporated in the revisable form for publishing.

`UBL-NDR-v3.0-csprd01.html`
An HTML rendering of the document.

`UBL-NDR-v3.0-csprd01.pdf`
A PDF rendering of the document.

These are the subdirectories in the package:

`art`
Diagrams and illustrations used in this specification

`db`
DocBook stylesheets for viewing in HTML the XML of this work product

Appendix B Acknowledgements (Non-Normative)

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Oriol Bausa Peris, Individual
Kenneth Bengtsson, Alfa1lab
Peter Borresen, Document Engineering Services Limited
Jon Bosak, Individual
Kees Duvekot, RFS Holland Holding B.V.
Martin Forsberg, Swedish Association of Local Authorities & Regions
G. Ken Holman, Crane Softwrights Ltd.
Ole Madsen, Danish Agency for Digitisation, Ministry of Finance
Tim McGrath, Document Engineering Services Limited
Andrew Schoka, Individual

Appendix C Summary of rules (Non-Normative)

The following is an alphabetical list of all rules expressed in this document. The title of each rule here is linked to the location in the document where the rule is presented, and the section number of the specification is indicated in parentheses. Only the first paragraph of each rule is copied here. Click on the rule title to jump to the rule. Click on the section number to jump to the section with the rule.

CCT01 CCTS Core Component Type schema (7.1.9)

The core component type schema of primitive data types for primary representation terms on which the unqualified data types are based shall be an unmodified copy of a schema fragment as published by UN/CEFACT.

COM01 CCTS information item description values (4.1.1)

Each item of describing information shall be a string value of Unicode characters without embedded hierarchical structure. The value itself may be structured in its syntax within the string.

COM02 CCTS information item prohibited characters and sequence (4.1.1)

All items of describing information shall be void of all sensitive XML markup characters and sequences. The following characters are prohibited in any CCTS information item:

COM03 Agreement on a list and rules of abbreviation in names (4.1.2)

The users of a CCTS model shall agree on a list (that may be empty) of words found in naming values that shall be abbreviated when a word is used in the component's derived name.

COM04 Agreement on a list and rules of abbreviation in model values (4.1.2)

The users of a CCTS model shall agree on a list (that may be empty) of accepted abbreviations allowed in values.

COM05 Agreement on a list of equivalences in names (4.1.2)

The users of a CCTS model shall agree on a list (that may be empty) of words found in naming values that shall be considered equivalent when a word is used in the component's derived name.

COM06 Leading name part in attribute names (4.1.2)

Every attribute's derived name shall be composed with the leading name part entirely in lower-case, even when that name part is an agreed-upon abbreviation.

COM07 Non-leading abbreviations in attribute names (4.1.2)

When an attribute's derived name is composed with an agreed-upon abbreviation in other than the leading name part, the abbreviation shall be entirely in upper-case.

COM08 CCTS information item name value prohibited characters (4.1.3)

All information items contributing to a component's dictionary entry name shall be void of all sensitive dictionary entry name markup characters.

COM09 Use of the singular form in names (4.1.3)
All information items contributing to a component's dictionary entry name shall be in the singular form unless the concept itself is plural.

COM10 Use of leading upper case letter in name value items (4.1.3)

COM11 Minimum set of values describing an ABIE component (4.1.4)
Each ABIE component of the model shall have at least a prescribed set of values required or available for definition.

COM12 Minimum set of values describing a BBIE component (4.1.4)
Each BBIE component of the model shall have at least a prescribed set of values required or available for definition.

COM13 Minimum set of values describing an ASBIE component (4.1.4)
Each ASBIE component of the model shall have at least a prescribed set of values required or available for definition.

COM14 ABIE contents cannot be empty (4.1.4)
Every ABIE shall contain at least one BIE.

DCL01 Element and type declarations (7.1.4)
Every element and type declaration shall be global.

DCL02 Library ABIE schema contents (7.1.4)
The one library ABIE schema shall include an element declaration for every library ABIE, every library ASBIE and every document ASBIE. It also includes a type declaration for every library ABIE.

DCL03 Document ABIE schema contents (7.1.4)
Each document ABIE schema shall include a single element declaration, that being for the document ABIE. It also includes a single type declaration, that being for the document ABIE.

DCL04 Library and document ABIE schema element declarations (7.1.4)
Each ABIE element shall be declared with the component name as the element name and the component name suffixed with "Type" as the type.

DCL05 Library and document ASBIE schema element declarations (7.1.4)
Each ASBIE element shall be declared with the component name as the element name and the component name of the ABIE of the associated object class suffixed with "Type" as the type.

DCL06 Library and document ABIE schema type declarations (7.1.4)
The library ABIE schema shall include a complex type declaration for every library ABIE. The document ABIE schema shall include a complex type declaration for the document ABIE.

DCL07 ABIE schema type content declaration order (7.1.4)

The members of an ABIE shall be ordered as a sequence with all BBIE element declarations first, followed by all ASBIE declarations.

DCL08 Document ABIE schema type extension declaration (7.1.4)

The members of a document ABIE that is supporting extensions shall be preceded by a reference to the extension collection element declaration.

DCL09 ABIE schema type content declaration style (7.1.4)

The members of an ABIE shall be declared using references to element declarations.

DCL10 Library BBIE schema (7.1.5)

The library BBIE schema shall have element declarations for every BBIE referenced in all document ABIEs and all library ABIEs.

DCL11 Library BBIE schema (7.1.5)

The library BBIE schema shall have a complex type declaration for every BBIE referenced in all document ABIEs and all library ABIEs.

DCL12 Library BBIE schema element declarations (7.1.5)

Each BBIE element shall be declared with the component name as the element name and the concatenation of the component name and "Type" as the type.

DCL13 Library BBIE schema element declarations (7.1.5)

Each BBIE element type shall be declared as simple content extended from a base of either a qualified data type or an unqualified data type without the addition of any additional attributes.

DTQ01 Data type qualifications for element content (7.1.10)

A CVA context shall be created for every BBIE with a non-empty value in the CCTS information for the qualified data type.

DTQ02 Data type qualifications for attribute content (7.1.10)

A CVA context shall be created for every CCTS supplementary component to be validated.

EXT01 Extension collection cardinality (7.1.6)

A document may be allowed to contain at most one extension collection, and if it is, that extension collection shall be optional and not repeated.

EXT02 Extension collection location (7.1.6)

The document's extension collection shall be realized in the validation artefacts as the first child of the document ABIE validation construct, before that of any BIE modeled for the document ABIE.

EXT03 Extension collection content (7.1.6)

The document's extension collection shall require one or more extension items as its content.

EXT04 Extension item content (7.1.6)

The extension item shall have a collection of metadata information regarding the given instance of the item, plus a mandatory single element being the extension point under which extension information is added to the document.

EXT05 Extension point (7.1.6)

The extension point of each extension item shall require a single element in some namespace other than the extension namespace. There shall be no other constraints imposed on the extension point element.

EXT06 Extension collection and item schema fragment and declarations (7.1.6)

The extension collection schema fragment shall include the declarations of the extension collection element, the extension item element, and the extension item metadata elements and any required type information for locally-defined elements.

EXT07 Extension point data type schema fragment and declaration (7.1.6)

The extension point schema fragment shall contain only a single complex type declaration being a sequence of exactly one element in a namespace other than the extension namespace to be processed with "lax validation".

EXT08 Extension point data type schema fragment import directives (7.1.6)

The extension point schema fragment may contain as many import directives for extension point schemas as desired. There is no order to the directives.

GEN01 Maintenance of information bundle composition (2.1)

There shall be a documented process by which information bundles are managed. Such a process shall support principles of the Core Component Technical Specification as described in other rules in this specification.

GEN02 Production of validation artefacts (2.1)

There shall be a documented process by which user data validation artefacts are created from the information maintained for information bundles.

MOD01 Information bundle structured as a document ABIE (3.1.1)

Each document shall be structured as a single top-level ABIE, referred to in this specification as a *document ABIE*.

MOD02 Libraries of available ABIEs (3.1.2)

Each library shall be structured as a collection of one or more ABIEs. Every collection shall not contain a document ABIE.

MOD03 Library ordering (3.1.2)

The ABIEs in a library shall be arranged in alphabetical order of the ABIE's dictionary entry name.

MOD04 Extension collection in each document information bundle (3.1.4)

Each information bundle shall allow for optional augmentation of a document with a collection of information not conceptually modeled by the document ABIE or the library

MOD04 Extension collection in each document information bundle (3.1.4)

ABIEs. There are no constraints on the information that may be included in an extension, including information concepts built using existing library components.

NAM01 Namespaces for information found in information bundles (7.1.1)

A minimum set of namespaces is required for that information that is found in information bundles.

NAM02 Namespaces for an extension (7.1.1)

Each extension shall have a set of namespaces defined.

NAM03 Namespaces for BBIE data types (7.1.1)

A minimum the following namespaces shall be defined for identifying type information for BBIEs.

NAM04 Namespaces for schema annotations (7.1.1)

When needed, a namespace for CCTS documentation found in schema annotations shall be defined as distinct from all other namespaces.

QDT01 Qualified Data Type declaration name (7.1.7)

Every qualified data type shall be declared using the name of the data type qualifier followed by the unqualified data type name.

QDT02 Qualified Data Type declaration basis (7.1.7)

Every qualified data type shall be based on an unqualified data type, imposing whatever qualifications are required to be expressed using XSD schema semantics.

UDT01 Unqualified Data Type declaration suite (7.1.8)

An unqualified data type shall be declared for every one of and only the 20 CCTS Primary Representation Terms and Secondary Representation Terms defined in Core Component Technical Specification Section 8-3 "Permissible Representation Terms".

UDT02 Unqualified Data Type declaration base (7.1.8)

Every unqualified data type shall be based on one of the 10 CCTS Core Component Type schema data types defined in Core Component Technical Specification Section 8-2 "Approved Core Component Type Content and Supplementary Components".

Appendix D Revision History (Non-Normative)

Revision	Date	Editor	Changes made
csd01wd01	06 January 2015	GKH	Initial version
csd01wd02	20 January 2015	GKH	Added examples; repaired COM13; added equivalences concept
csd01wd03	23 January 2015	GKH	Repaired prose derived expressions and included formula examples
csd01wd04	2 February 2015	GKH	Copyfit code fragments; improve grammar
csd01	4 February 2015	GKH	Date, stage and status of release
csprd01	4 February 2015	GKH	Public review draft