XML Naming and Design Rules

Draft 1.0, 3 August 2004

This UN/CEFACT – XML Naming and Design Rules has been developed in accordance with the UN/CEFACT/TRADE/22 Open Development Process (ODP) for Technical Specifications. It has been approved by the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) Applied Techniques Group (ATG) for promulgation for public review in accordance with Step 5 of the ODP.

Distribution of this document is unlimited.

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

This version:

UN/CEFACT – XML Naming and Design Rules for Core Components, Version 1.0 of 3 August 2004

Previous version:

NamingAndDesignRules_0p4r1.doc

Document identifier:

NamingAndDesignRules_1.0.doc

Location:

http://www.disa.org/cefact-groups/atg/downloads/index.cfm
UN/CEFACT – XML Naming and Design Rules Project
Team Participants

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

Project Team Leader:
Mark Crawford    LMI

Editors:
Gunther Stuhec    SAP AG
Paula Heilig    Worldspan
Margaret Pemberton    Diskray

Contributors:
Hisanao Sugamata    ECOM-Japan
Frank Lin    GCOM
K.K. Suen    EAN Hong Kong
Luc Mouchot    CNAM-TS
Thomas Bikeev    EAN.UCC
Jostein Frømyr    EDISYS
Sue Probert    SEPIA eb

Acknowledgements

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

SWIFT
U.S. Department of the Navy
U.S. Environmental Protection Agency
U.S. Federal CIO Council XML Working Group
OpenTravel Alliance
Australia Electricity & Gas Industry
CIDX
EAN/UCC
European Transmission System Operators
PIDX
# Table of Contents

1 INTRODUCTION ................................................................................................................................. 7  
  1.1 Scope and Focus ................................................................................................................................. 7  
  1.2 Audiences .......................................................................................................................................... 7  
  1.3 Structure of this Specification ........................................................................................................... 7  
  1.4 Terminology and Notation ................................................................................................................ 7  
  1.5 Related Documents ............................................................................................................................ 8  
  1.6 Conformance ....................................................................................................................................... 8  
  1.7 Guiding Principles ............................................................................................................................... 8  

2 GENERAL XML CONSTRUCT ............................................................................................................. 9  
  2.1 Overall Schema Structure ................................................................................................................ 9  
  2.2 Relationship to the CCTS .................................................................................................................. 9  
  2.3 Naming and Modeling Constraints .................................................................................................... 12  
    2.3.1 Naming Constraints ....................................................................................................................... 12  
    2.3.2 Modelling Constraints .................................................................................................................. 14  
  2.4 Reusability Scheme ............................................................................................................................ 14  
    2.4.1 Element Naming Conventions ..................................................................................................... 14  

2.5 Modularity Model ............................................................................................................................... 15  
  2.5.1 Root Schema .................................................................................................................................... 15  
  2.5.2 Internal Schema .............................................................................................................................. 15  
  2.5.3 External Schema ............................................................................................................................. 16  
    2.5.3.1 Core Component Type Schema Module .................................................................................... 16  
    2.5.3.2 Unqualified Data Type Schema Module .................................................................................... 17  
    2.5.3.3 Qualified Data Type Schema Module ....................................................................................... 17  
    2.5.3.4 Reusable Aggregate Business Information Entity Schema Module ...................................... 17  
    2.5.3.5 Code List Schema Modules ...................................................................................................... 17  
    2.5.3.6 Identifier List Schema Modules ............................................................................................... 18  
    2.5.3.7 External Standards Body Aggregate Business Information Entity Schema Modules ................ 18  

2.6 Namespace Scheme ............................................................................................................................ 18  
  2.6.1 UN/CEFACT Namespace Scheme ................................................................................................. 19  
  2.6.2 Declaring Namespace ..................................................................................................................... 20  
  2.6.3 Namespace Persistence ................................................................................................................... 21  
  2.6.4 Namespace Uniform Resource Identifiers ..................................................................................... 21  
  2.6.5 Namespace Constraint ................................................................................................................... 22  
  2.6.6 UN/CEFACT Schema Namespace Tokens .................................................................................... 22  

2.7 Schema Location .................................................................................................................................. 22  

2.8 Versioning ............................................................................................................................................ 23
4. INCLUDES....................................................................................................................................... 71
5. IMPORTS......................................................................................................................................... 72
6. ROOT ELEMENT ............................................................................................................................ 73
7. TYPE DEFINITIONS........................................................................................................................ 74
APPENDIX B. ANNOTATION TEMPLATES............................................................................................. 78
APPENDIX C. NAMING & DESIGN RULES LIST .................................................................................... 81
APPENDIX D. DEFINITION OF TERMS ................................................................................................... 93
APPENDIX E. RELATED DOCUMENTS .................................................................................................. 96
APPENDIX F. CORE COMPONENT SCHEMA MODULE........................................................................ 98
APPENDIX G. UNQUALIFIED DATA TYPE SCHEMA MODULE.......................................................... 110
1 Introduction

This UN/CEFACT – XML Naming and Design Rules describes and specifies the rules and guidelines that will be applied by UN/CEFACT when developing XML schema specifications. This UN/CEFACT – XML Naming and Design Rules provides a way to identify, capture and maximize the re-use of business information expressed as XML schema components to support and enhance information interoperability across multiple business situations.

1.1 Scope and Focus

This UN/CEFACT – XML Naming and Design Rules can be employed wherever business information is being shared or exchanged amongst and between enterprises, governmental agencies, and/or other organisations in an open and worldwide environment using the XML schema language for defining the content of the information exchange. This specification will form the basis for standards development work of technical experts developing XML schema specifications based on information models developed in accordance with the UN/CEFACT – Core Components Technical Specification.

1.2 Audiences

The primary audience for this UN/CEFACT – XML Naming and Design Rules are member of the UN/CEFACT Applied Technologies Group, working group on XML who are responsible for development and maintenance of the UN/CEFACT XML schema specifications and the wider membership of the other UN/CEFACT Groups who will participate in the process of creating and maintaining the UN/CEFACT XML schema specifications. Additional audiences are designers of tools who need to specify the conversion of user input in the tool into the XML schema representation adhering to the roles defined in this document as well as designers of XML schema specifications outside of the UN/CEFACT Forum community.

1.3 Structure of this Specification

These UN/CEFACT – XML Naming and Design Rules have been divided in to 5 main sections. Section 1 provides general information about the document itself. Section 2 provides information on the guiding principles applied in developing this specification as well as its dependency and relationship to the UN/CEFACT – Core Components Technical Specification. Furthermore this section describes the approach taken to modularity in order to maximize the re-use of business information expressed as XML schema components and the general naming conventions applied. Section 3 provides the general conventions applied with respect to the use of the XML schema language. Section 4 provides detailed rules applicable to each of the schema modules defined by the modularity approach. Section 5 provides guidelines and rules related to XML instance documents.

1.4 Terminology and Notation

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in Internet Engineering Task Force (IETF) Request For Comments (RFC) 2119.1 [Definition] – A formal definition of a term. Definitions are normative. [Example] – A representation of a definition or a rule. Examples are informative.

1.5 Related Documents

The documents referenced in this document are listed in informative Appendix E.

1.6 Conformance

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

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

1.7 Guiding Principles

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

- Relationship to UMM – UN/CEFACT XML Schemas will be based on UMM metamodel adherant Business Process Models.
- Relationship to Information Models – UN/CEFACT XML Schemas will be based on UML-based Information Models.
- Schema Creation – UN/CEFACT XML design rules will support schema creation through handcrafting as well as automatic generation from UML profiled objects and XML profiled objects.
- ebXML Use – UN/CEFACT XML Schemas and instance documents shall be straightforwardly usable within the ebXML framework and compatible with other frameworks to the maximum extent practicable.
- Interchange and Application Use – UN/CEFACT XML Schemas and instance documents are intended for business-to-business and application-to-application use.
- Tool Use and Support - The design of UN/CEFACT XML Schemas will not make any assumptions about sophisticated tools for creation, management, storage, or presentation being available.
- Time Constraints – Urgency is a key item in the development of UN/CEFACT XML design rules.
- Legibility - UN/CEFACT XML instance documents should be intuitive and reasonably clear in the context for which they are designed.
- Schema Features - The design of UN/CEFACT XML Schemas should use the most commonly supported features of W3C XSD Schema.
- Technical Specifications – UN/CEFACT XML design rules will be based on Technical Specifications holding the equivalent of W3C recommended status.
- Schema Specification – UN/CEFACT XML design rules will be fully conformant with W3C XML Schema Definition Language.
- Interoperability - The number of ways to express the same information in a UN/CEFACT XML Schema and UN/CEFACT XML instance document is to be kept as close to one as possible.
- Customization - The design of UN/CEFACT XML Schemas must facilitate customization.
- Maintenance – The design of UN/CEFACT XML Schemas must facilitate maintenance.
- Context Sensitivity - The design of UN/CEFACT XML Schemas must ensure that context-sensitive document types aren’t precluded.
- Relationship to Other Namespaces - UN/CEFACT XML design rules will be cautious about making dependencies on other namespaces.
- Legacy formats - UN/CEFACT XML design rules are not responsible for sustaining legacy formats.
- Messages must express semantics fully in schemas and not rely on well-formedness.
2 General XML Construct

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

- Overall Schema Structure
- Relationship to CCTS
- Naming and Modeling Constraints
- Reusability Scheme
- Modularity Strategy
- Namespace Scheme
- Versioning Scheme
- Schema Documentation Requirements

2.1 Overall Schema Structure

To maintain consistency in lexical form, all UN/CEFACT schema need to use a standard structure for all content. This standard structure is contained in Appendix A.

[R 2] UN/CEFACT Schema MUST follow the standard structure defined in Appendix A.

2.2 Relationship to the CCTS

All UN/CEFACT business modeling and business process definition employs the methodology and model described in Core Components Technical Specification, Part 8 of the ebXML Technical Framework (CCTS). The CCTS is a continuation of work that originated in, and remains a part of, the ebXML initiative. CCTS defines a new paradigm in data definition and use. CCTS defines context neutral and context specific building blocks. Context neutral components are defined as Core Components (ccts:CoreComponents). Context neutral ccts:CoreComponents are defined in CCTS as "A building block for the creation of a semantically correct and meaningful information exchange package. It contains only the information pieces necessary to describe a specific concept." Figure 2-1 illustrates the various pieces of the overall ccts:CoreComponents metamodel.

---

The context specific components are defined as Business Information Entities (ccts:BusinessInformationEntities). Context specific ccts:Business InformationEntities are defined in CCTS as “A piece of business data or a group of pieces of business data with a unique Business Semantic definition.” Figure 2-2 illustrates the various pieces of

---

3 See CCTS Section 6.2 for a detailed discussion of the ebXML context mechanism.
the overall ccts:BusinessInformationEntity metamodel and their relationship with the ccts:CoreComponents metamodel.

Figure 2-2 Context Specific Business Information Entity Metamodel

UN/CEFACT XML design rules will be closely coupled with CCTS. UN/CEFACT Schemas will be developed from fully conformant Business Information Entities that are based on fully conformant Core Components. Figure 2-3 shows the relationship between CC’s, BIE’s and XSD artefacts. The gray boxes reflect CCTS constructs (CCTs, DTs, CCs, BIEs), and the white and yellow boxes reflect XSD constructs (xsd:types, xsd:elements, xsd:attributes). The relationships follow the following basic principles:
2.3 Naming and Modeling Constraints

UN/CEFACT XML is derived from CCTS and UN/CEFACT Modelling Methodology (UMM) process modelling and data analysis as precursors to developing the UN/CEFACT XML component library. In determining how best to affect this work, several constraints have been identified that directly impact on both the process modelling and data analysis, as well as on the resultant UN/CEFACT XML Schema.

2.3.1 Naming Constraints

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

[R 3] Each element or attribute XML name MUST have one and only one fully qualified XPath (FQXP).
This rule and the other rules on element naming imply that the fully qualified XPath will always represent the CCTS dictionary entry name of the corresponding ABIE, BBIE, ASBIE or DT.

Example 2-1:

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

[R 4] Element, attribute and type names MUST be in the English language, using the primary English spellings provided in the *Oxford English Dictionary*.

The official language for UN/CEFACT XML is English. All official XML constructs as published by UN/CEFACT will be in English. *XML development work* may very well occur in other languages, however official submissions for inclusion in the UN/CEFACT XML library must be in English. Other language translations of UN/CEFACT published XML components are at the discretion of users. Strict literal translations for implementation and run-time use are considered fully compliant with the published XML component.

Example 2-2: Attribute

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

[R 5] Lower-camel-case (LCC) MUST be used for naming attributes.

Example 2-3: Element

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

[R 6] Upper-camel-case (UCC) MUST be used for naming elements and types.

Example 2-4: Type

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

[R 7] Names MUST be in singular form unless the concept itself is plural.

Example 2-5: Singular and Plural Concept Form

**Singular:**

```
<xsd:element name="ItemQuantity" .../>
```

**Plural:**

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

[R 8] Names MUST NOT contain non-letter characters, unless required by language-specific rules.

Example 2-6: Non-Letter Characters

**Not Allowed**

```
<xsd:element name="LanguageCode8" .../>
```
XML names constructed from dictionary entry names MUST NOT include periods, spaces, or other separators; or characters not allowed by W3C XML 1.0 for XML names.

Not Allowed

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

Element and `xsd:simple` and `xsd:complexType` names MUST NOT use acronyms, abbreviations, or other word truncations, except those included in the UN/CEFACT controlled vocabulary.

### 2.3.2 Modelling Constraints

UN/CEFACT has developed a robust modelling methodology (UMM). This modelling methodology forms the basis for all UN/CEFACT business process analysis conducted by, and process models developed by, UN/CEFACT TBG. In keeping with our guiding principles, modelling constraints are limited to those necessary to ensure consistency in development.

### 2.4 Reusability Scheme

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

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

### 2.4.1 Element Naming Conventions

The fully qualified path anchors the use of that construct to a particular location in a business message. The dictionary definition identifies any semantic dependencies that the FQXP has on other elements and attributes within the UN/CEFACT library that are not otherwise enforced or made explicit in its structural definition. The dictionary serves as a traditional data dictionary, and also serves some of the functions of traditional implementation guides. As discussed in Section 2.4 above, the dictionary must be carefully controlled to overcome the limitations in control inherent in a local element approach.

Each element name declaration MUST be based on the property term and qualifiers and the representation term of the BBIE or the property term and object class of the ASBIE. If there are successive duplicate words in the property term and representation terms of the source dictionary entry name which represent exactly the same semantics, then the duplicate words MUST be removed.
2.5 Modularity Model

Modularity in schema design promotes reuse and provides significant management capabilities. Modules can be either unique in their functionality, or represent splitting of larger schema files for performance or manageability enhancement. A modularity model provides an efficient and effective mechanism for importing components as needed rather than dealing with complex, multi-focused schema.

2.5.1 Root Schema

UN/CEFACT incorporates a modularity concept that leverages the benefits previously described. In the UN/CEFACT XML repository, there are a number of UN/CEFACT root schemas, each of which expresses a separate business function. Each root schema will consist of a single named complex type that defines associations to xsd constructs contained in internal (same namespace) or external (different namespace) schema modules.

[R 13] A root schema MUST be created for each unique business information exchange.

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

[R 14] A root schema MUST NOT replicate reusable constructs available in schema modules capable of being referenced through xsd:include or xsd:import.

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

[R 15] UN/CEFACT schema modules MUST either be treated as external schema modules or as internal schema modules of the root schema.

2.5.2 Internal Schema

For larger namespaces, schema modules – internal schema modules – may be defined. UN/CEFACT schemas may have zero or more internal modules that they include. Root schemas are limited to defining a single complex type that fully describes the business information exchange.

[R 16] Root schemas MUST only define a single complex type that fully describes the business information exchange.

If additional complex types are needed that don’t properly belong in the reusable ABIE module (See Section 2.5.5 below), then those complex types will be defined in an internal schema module. Internal schema modules will reside in the same namespace as their parent root schema. Since the internal schema reside in the same namespace as the root, the root schema uses xsd:include to incorporate these internal modules. The UN/CEFACT schema modularity model ensures that logical associations exist between root and internal schema modules and that individual modules can be reused to the maximum extent possible.

[R 17] All UN/CEFACT internal schema modules MUST be in the same namespace as their corresponding rsm:RootSchema.

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

[R 18] Each UN/CEFACT internal schema module MUST be named

\{ParentRootSchemaModuleName\}{InternalSchemaModuleNameFunction}\{Schema Module\}
2.5.3 External Schema

To adhere to the principles and rules contained in Section 2.5.1, schema modules will be created for reusable components. External schema modules will be used by several root schemas. The root schema may import one or more of these external schema modules. UN/CEFACT has identified the need for the following external schema modules:

- Core Component Types
- Unqualified Data Types
- Qualified Data Types
- Reusable ABIEs
- Code Lists
- Identifier Lists (when the functionality mimics that of a code list)
- Other External Standard Body ABIE modules

These external schema modules are reflected in Figure 2-4.

![Figure 2-4 UN/CEFACT Schema Modules](image)

2.5.3.1 Core Component Type Schema Module

A schema module is required to represent the normative form for CCTs from CCTS. This schema module will be used as the normative reference for all CCTS based XML instantiations.

[R 19] A core component type schema module MUST be created

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

[R 20] The \texttt{cct:CoreComponentType} schema module MUST be named "CCTS CCT Schema Module"

---

5 The terms “unqualified data type” and “qualified data type” refer to the ISO 11179 concept of qualifiers for name constructs, not to the xml namespace concept of qualified and unqualified.
2.5.3.2 Unqualified Data Type Schema Module

A single unqualified data type schema module is required. This schema module will contain a type
definition for each data type expressed by approved CCTS primary and secondary representation terms.

[R 21] An unqualified data type schema module MUST be created

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

[R 22] The udt:UnqualifiedDataType schema module MUST be named "UN/CEFACT Unqualified
Data Type Schema Module"

2.5.3.3 Qualified Data Type Schema Module

A single qualified data type schema module is required. This schema module will contain a type definition
for each qualified data type as defined in CCTS.

[R 23] A qualified data type schema module MUST be created

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

[R 24] The qdt:QualifiedDataType schema module MUST be named "UN/CEFACT Qualified Data
Type Schema Module"

2.5.3.4 Reusable Aggregate Business Information Entity Schema Module

A single reusable aggregate business information entity schema module is required. This schema
module will contain a type definition for every reusable ABIE in the UN/CEFACT Core Component
Library.

[R 25] An aggregate business information entity schema module MUST be created

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

[R 26] The ram:ReusableAggregateBusinessInformationEntity schema module MUST be
named "UN/CEFACT Aggregate Business Information Entity Schema Module"

2.5.3.5 Code List Schema Modules

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

[R 27] Reusable code list schema modules MUST be created to convey code list enumerations

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

[R 28] The name of each clt:CodeList schema module MUST be of the form: <Code List Agency
Name><Code List Name> - Code List Schema Module

Where:
Code List Agency Name = Agency that maintains the code list
Code List Name = The name of the code list as assigned by the agency that maintains the code
list

Example 2-7:

UN/CEFACT Account Type Code - Code List Schema Module
2.5.3.6 Identifier List Schema Modules

In those cases where identifier schemes mimic the functionality of code lists, reusable identifier lists schema modules will be created to minimize the impact of identifier list changes on root and other reusable schema. Each reusable identifier list schema module will contain enumeration values for codes and code values.

[R 29] An Identifier List schema module MUST be created for each identifier list that mimics code list functionality to convey enumerations of the identifier list value and token for that value.

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

[R 30] The name of each UN/CEFACT identifier list schema module MUST be of the form: <Identification Scheme Agency Name><Identification Scheme Name> - Identifier List Schema Module

Where:
Identification Scheme Agency Name = Agency that maintains the identifier list
Identification Scheme Name = Name as assigned by the agency that maintains the identifier list

Example 2-8:

ISO Country Identifier - Identifier List Schema Module

2.5.3.7 External Standards Body Aggregate Business Information Entity Schema Modules

The external Standards Body ABIE modules are those reusable XML constructs created by other standards bodies and made available for reuse. UN/CEFACT will only import external ABIE modules when their contents are in strict conformance to the requirements of the CCTS and this document.

[R 31] Imported schema modules MUST be fully conformant with UN/CEFACT naming and design rules and the Core Components Technical Specification.

2.6 Namespace Scheme

As defined in the W3C specification, “XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references.” This enables interoperability and consistency in the XML artifacts for the extensive library of reusable types and schema modules. The UN/CEFACT reusability approach that maximizes reuse of defined named types and locally declared elements and attributes within those types (See Section 2.4) and modularity approach of multiple reusable schema modules (See Section 2.5) proscribe just such an approach. There exists specific relationships between the various internal and external schema modules identified in Section 2.8.3 with respect to their namespaces. These relationships are defined in Figure 2.5. Accordingly, a sufficiently robust namespace scheme is essential.

---

6 World Wide Web Consortium, Namespaces in XML, 14 January 1999
In establishing a UN/CEFACT approach to namespaces, it is important to recognize that in addition to XML requirements, many other requirements exist for a standardized namespace approach. Accordingly, an overarching UN/CEFACT namespace scheme must be sufficiently flexible and robust to accommodate both XML and other syntax requirements. Figure 2-6 reflects such an approach and will be used as the basis for determining the namespace structure and rules that follow.
2.6.2 Declaring Namespace

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

[R 32] Every UN/CEFACT defined or used schema module MUST have a namespace declared, using the `xsd:targetNamespace` attribute.
2.6.3 Namespace Persistence

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

[R 33] Every defined or used schema module version other than internal schema modules MUST have its own unique namespace.

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

[R 34] UN/CEFACT published namespace declarations or contents MUST never be changed.

2.6.4 Namespace Uniform Resource Identifiers

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

[R 35] UN/CEFACT namespaces MUST be defined as URNs

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

\[
<URN> ::= \text{urn:} \langle \text{NID} \rangle \text{:} \langle \text{NSS} \rangle
\]

where:

- \langle \text{NID} \rangle = \text{the Namespace Identifier}
- \langle \text{NSS} \rangle = \text{the Namespace Specific String},
  The leading "urn:" sequence is case-insensitive.
- The Namespace ID determines the syntactic interpretation of the Namespace Specific String

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

\[
\]

Where:

- Namespace Identifier (NID) = UN
- Namespace Specific String = unece:uncefact:<schematype>:<status>:<name>:<major>:<minor>:[<revision>] with unece and uncefact as fixed value second and third level domains within the NID of un
- schematype = a token identifying the type of schema module: data|process|codelist
- status = the status of the schema as: draft|standard
- name = the name of the module (using underscore as separator)
- major = The major version number. Sequentially assigned, first release starting with the number 1.
2.6.5 Namespace Constraint

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

[R 38] UN/CEFACT namespaces MUST only contain UN/CEFACT developed schema modules.

2.6.6 UN/CEFACT Schema Namespace Tokens

A unique token will be defined for each namespace. The exact token for each type of namespace will be defined by the applicable schema module subsection in Section 4.

2.7 Schema Location

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

\['urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>:<minor>:<revision>\]

[R 39] The general structure for schema location MUST be:

http://www.unece.org/uncefact/<schematype>/<name>_<major>_<minor>_<revision>.xsd

Where:
schematype = a token identifying the type of schema module: data|process|codelist
name = the name of the module (using underscore as separator)
major = the major version number, sequentially assigned, first release starting with the number 1.
minor = the minor version number within a major release, sequentially assigned, first release
starting with the number 0.
revision = sequentially assigned alphanumeric character for each revision of a minor release.
status = the status of the schema as: draft|standard

[R 40] Each \texttt{xsd:schemaLocation} attribute declaration MUST contain a persistent and resolvable
URL.

[R 41] Each \texttt{xsd:schemaLocation} attribute declaration URL MUST contain an absolute path.

[R 42] Schema modules MUST be located under the directory:

http://www.unece.org/uncefact/cc/schema/<schema-mod-name>.xsd

2.8 Versioning

[R 43] A UN/CEFACT namespace URN is divided into three parts. First, is the standard UN/CEFACT
namespace information. Second, is the description of the purpose of the namespace. Third, is the
version information. The version information will in turn be divided into major (or incompatible)
and minor (or compatible) fields. The minor field has an optional revision extension.

2.8.1 Major Versions

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

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

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

\['urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>:0\]

Where:

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

[R 44] Every schema major version MUST have the URI of:

\['urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>:0:<revision>\]

Major version numbers should be based on logical progressions to ensure semantic understanding of the
approach and guarantee consistency in representation. Non-negative, sequentially assigned incremental
integers satisfy this requirement.
Every UN/CEFACT schema and schema module major version number MUST be a sequentially assigned incremental integer greater than zero.

2.8.2 Minor Versions

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

- Restrictions of types, but not to required elements
- Adding values to enumerations
- Optional extensions
- Restrictions on facets
- Delete or add optional elements
- Change cardinality from optional to mandatory

Minor versioning MUST be limited to declaring new optional xsd constructs, extending existing xsd constructs and refinements of an optional nature.

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

```
urn:un:unece:uncefact:<schematype>:<status>:<name>:<major-number>:non-zero:[<revision>]  
```

Where:

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

Every UN/CEFACT schema minor version MUST have the URI of:

```
urn:un:unece:uncefact:cc:schema:<name>:<major-number>:<non-zero integer>:[<revision>]  
```

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

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

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

UN/CEFACT minor version schema MUST incorporate all XML constructs from the immediately preceding major or minor version schema.

[Note]

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


[R 52] All schemata and instances MUST be based on the W3C suite of technical specifications holding recommendation status.

3.1 Schema Construct

[R 53] Element form default MUST be declared as qualified.

[R 54] Attribute form default MUST be declared as unqualified.

Example 3-1: Element and Attribute Form Default

```xml
<xsd:schema targetNamespace="... see namespace ...
xmlns:xsd="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified" attributeFormDefault="unqualified">
```

3.1.1 Constraints on Schema Construction

[R 55] The "xsd" prefix MUST be used.

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

[R 56] The XSI prefix SHALL be used where appropriate.

[R 57] Processing Instructions MUST NOT be used.

[R 58] Notations MUST NOT be used.

[R 59] Wildcards MUST NOT be used.

[R 60] The `xsd:any` element MUST NOT be used.

[R 61] The `xsd:any` attribute MUST NOT be used.

[R 62] Mixed content MUST NOT be used (excluding documentation).

[R 63] Substitution groups MUST NOT be used.

[R 64] ID/IDREF MUST NOT be used.

[R 65] Key/KeyRef MUST be used for information association.

[R 66] The absence of a construct or data MUST NOT carry meaning.
3.2 Attribute and Element Declarations

3.2.1 Attributes

3.2.1.1 Usage of Attributes

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

[R 67] User defined attributes MUST only be used to convey core component type (CCT) supplementary component information.

3.2.1.2 Constraints on Attribute Declarations

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

[R 68] The nillable attribute MUST NOT be used.

3.2.2 Elements

3.2.2.1 Usage of Elements

Elements are used to define the top-level complex type that represents business process, all attributes of an object class (BBIE and ASBIE), and the object class itself (ABIE).

3.2.2.2 Element Declaration

[R 69] All element declarations MUST be local except for a root element that must be declared globally.

[R 70] Empty elements MUST NOT be used.

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

[R 71] The xsd: type of each leaf element declaration MUST be of the data type of its source business information entity (BBIE) or complex type of its source association business information entity (ASBIE).

Example 3-2:

```xml
<xsd:complexType name="AccountType">
  <xsd:annotation>
    ...see annotation...
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="ID" type="udt:IdentifierType"
      minOccurs="0" maxOccurs="unbounded">
    </xsd:element>
    <xsd:element name="Status" type="ram:StatusType"
      minOccurs="0" maxOccurs="unbounded">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```
3.2.2.3 Constraints on Element Declarations


[R 73] The xsd:all element MUST NOT be used.

[R 74] Mixed-content elements MUST NOT be used.

3.3 Type Definitions

3.3.1 Usage of Types

[R 75] All type definitions MUST be named.

Example 3-3:

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

[R 76] Type definitions MUST NOT duplicate the functionality of existing xsd:built-in simple types.

3.3.2 Simple Type Definitions

[R 77] User defined simple types definitions MUST NOT be used.

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

Example 3-4: Simple Types in Unqualified Data Types Schema Module

```xml
<xsd:simpleType name="DateTimeType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
```
Example 3-5: Simple Types in Code Lists Module

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

3.3.3 Complex Type Definitions

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

Example 3-6: Complex Type of Object Class “AccountType”

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

3.4 Use of XSD Extension and Restriction

The general philosophy is that all UN/CEFACT schema constructs will follow the model defined in Figure 2.1. These schema constructs are based on the concept that the underlying semantic structures of the core components and business information entities are normative forms of standards that developers are not allowed to alter without coordination of appropriate TBG groups (including TBG17 - Harmonization) and ICG. Accordingly, as business requirements dictate, new schema constructs will be created and new types defined and elements declared as appropriate and the concept of derivation through the use of xsd extension and restriction will only be used in limited circumstances as described below.

3.4.1 XSD Extension

[R 78] Extension MUST only be used in the cct:CoreComponentTypes schema module and the udt:UnqualifiedDataType schema module. When used it MUST only extend a built-in xsd:datatype.

3.4.2 Restriction

The CCTS specification employs the concept of semantic restriction in creating specific instantiations of core components. Accordingly, xsd:restriction will be used as appropriate to define types that are derived from the existing types. Where used, the derived types must always be renamed. Simple and complex type restrictions may be used.
Example 3-7: Restriction of Simple Type

```xml
<xsd:simpleType name="IndicatorType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:restriction base="xsd:boolean">
    <xsd:pattern value="false"/>
    <xsd:pattern value="true"/>
  </xsd:restriction>
</xsd:simpleType>
```

### 3.5 Annotation

In the UN/CEFACT schema modules the `xml:annotations` will only be used to provide documentation.

#### 3.5.1 Documentation

The annotation documentation will be used to convey all metadata as specified in the CCTS, i.e., to convey the semantic content carried in the XML construct. The following sets of annotations are required as defined in section 4 in type definitions and element declarations (the representation of each item in XML code is shown in parens):

- **Unique Identifier**: The unique identifier assigned to the artefact in the library. (`UniqueID`)
- **Category Code**: The category to which the artefact belongs. (`CategoryCode`)
- **Dictionary Entry Name**: The complete name (not the tag name) of the artefact in the library. (`DictionaryEntryName`)
- **Cardinality**: An indication of whether the property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the object. (`CardinalityText`)
- **Definition**: The semantic meaning of the artefact. (`DefinitionText`)
- **Object Class**: The Object Class represented by the artefact. (`ObjectClassTermName`)
- **Property Term**: The Property Term represented by the artefact. (`PropertyTermName`)
- **Representation Term**: The Representation Term represented by the artefact. (`RepresentationTermName`)
- **Associated Object Class Term**: The Associated Object Class Term represented by the artefact. (`AssociatedObjectClassTermName`)
- **Qualifier Term**: A term(s) that qualifies the Object Class and/or Property. (`QualifierTermName`)
- **Business Process Context**: A valid value describing the Business Process contexts for which this construct has been designed. Default is “In All Contexts”. (`BusinessProcessContext`)
- **Geopolitical/Region Context**: A valid value describing the Geopolitical/Region contexts for which this construct has been designed. Default is “In All Contexts”. (`GeopoliticalOrRegionContext`)
- **Official Constraints Context**: A valid value describing the Official Constraints contexts for which this construct has been designed. Default is “None”. (`OfficialConstraintContext`)
- **Product Context**: A valid value describing the Product contexts for which this construct has been designed. Default is “In All Contexts”. (`ProductContext`)
- **Industry Context**: A valid value describing the Industry contexts for which this construct has been designed. Default is “In All Contexts”. (`IndustryContext`)
- **Role Context**: A valid value describing the Role contexts for which this construct has been designed. Default is “In All Contexts”. (`BusinessProcessRoleContext`)
- **Supporting Role Context**: A valid value describing the Supporting Role contexts for which this construct has been designed. Default is “In All Contexts”. (`SupportingRoleContext`)
- **System Capabilities Context**: A valid value describing the Systems Capabilities contexts for which this construct has been designed. Default is “In All Contexts”. (`SystemCapabilitiesContext`)
- **Usage Rule**: A constraint that describes specific conditions which are applicable to the artefact. (`UsageRuleText`)
- **Business Term**: A synonym term under which the artefact is commonly known and used in business. (`BusinessTermName`)
Appendix B specifies normative information on the specific annotation required for each of the artefacts.

Example 3-8: Example of annotation

```xml
<xsd:annotation xml:lang="en">
  <xsd:documentation>
    <ccts:UniqueID>UN00000002</ccts:UniqueID>
    <ccts:CategoryCode>BBIE</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Account Identifier</ccts:DictionaryEntryName>
    <ccts:Definition>The identification of a specific account.</ccts:Definition>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
    <ccts:PropertyTermName>Identifier</ccts:PropertyTermName>
    <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
    <ccts:BusinessTermName>Account Number</ccts:BusinessTermName>
  </xsd:documentation>
</xsd:annotation>
```

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

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

4.1 Root Schema

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

4.1.1 Schema Construct

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

Example 4-1: Structure of RootSchema Module

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Gunther Stuhec (UBL) -->
<!-- ====================================================================== -->
<!-- ===== [MODULENAME] Schema Module; [VERSION] ===== -->
<!-- ====================================================================== -->
<!-- Module of [MODULENAME], Agency: UN/CEFACT, Version: 0.3 Rev. 6 Last change: 25. June 2004 -->
<!-- Copyright (C) UN/CEFACT (2004). All Rights Reserved. -->
<!-- see copyright information ... -->
<xsd:schema
    targetNamespace="urn:un:unece:uncefact:data:draft:[MODULENAME]:0:3:6"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- ===== Includes ===== -->
    <!-- ====================================================================== -->
    <!-- ===== Include of [MODULENAME] ===== -->
    <!-- see includes ... -->
    <!-- ===== Imports ===== -->
    <!-- ====================================================================== -->
    <!-- ===== Import of [MODULENAME] ===== -->
    <!-- see imports ... -->
    <!-- ===== Root Element ===== -->
    <!-- see root element declaration ... -->
    <!-- ===== Type Definitions ===== -->
    <!-- see type definition ... -->
</xsd:schema>
```

4.1.2 Namespace Scheme

[R 79] The root schema module MUST be represented by the token “rsm”.

NamingAndDesignRules_1.0.doc Page 32  8/12/2004
4.1.3 Imports and Includes

[R 80] The rsm:RootSchema MUST import the following schema modules:
- ram:ReusableABIE Schema Module
- udt:UnqualifiedDataTypes Module
- qdt:QualifiedDataTypes Module

The root schema may import other external schema modules as necessary provided they conform to UN/CEFACT naming and design rules.

One root schema (root schema A) may also make use of ABIEs defined as part of another root schema (root schema B) or that root schemas internal schema module, i.e. type definitions and element declarations defined in another namespace. An example may be that the root schema for an Order Response message (root schema A) makes use of ABIEs defined as part of the schema definition for an Orders message (root schema B). If that is the case then such type definitions and element declarations should be imported in to the root schema (root schema A). To achieve this only the root schema (root schema B) in the namespace containing the type definitions and element declarations needed should be imported as this in itself included the subordinate internal schema modules.

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

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

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

4.1.4 Root Element Declaration

Each UN/CEFACT business message has a single root element that is globally declared in the root schema representing the business information exchange. The root element is named according to the business information exchange that it represents and references the message assembly that contains the actual business information.
A single global element known as the root element MUST be globally declared in a `rsm:RootSchema`.

The root element MUST reflect the Message Assembly that is defined for the actual content of the business information.

The name of the root element MUST be the name of the Message Assembly with separators and spaces removed.

Example 4-3:

```xml
<!-- ===== Root Element ===== -->
<!-- ============================ -->
<xsd:element name="PurchaseOrder" type="exp:PurchaseOrderType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:element>
```

4.1.5 Type Definitions

For the root element a corresponding complex type that represents the Message Assembly MUST be defined.

The name of the top-level complex type MUST be the name of the root element with the word "type" appended.

4.1.6 Annotations

For every `rsm:RootSchema` root element declaration a structured set of annotations MUST be present in the following pattern:

- **UniqueID** (mandatory): The identifier that references the Message Assembly instance in a unique and unambiguous way.
- **CategoryCode** (mandatory): The category to which the object belongs. In this case the value will always be RSM.
- **Name** (mandatory): The name of the Message Assembly
- **VersionID** (mandatory): An indication of the evolution over time of a Message Assembly.
- **Description** (mandatory): A brief description of the business information exchange.
- **BusinessDomain** (mandatory, repetitive): The TBG group(s) that developed this Message Assembly.
- **BusinessProcessContext** (mandatory, repetitive): The business process with which this Message Assembly is associated.
- **GeopoliticalorRegionContext** (optional, repetitive): The geopolitical/region contexts for this Message Assembly.
- **OfficialConstraintContext** (optional, repetitive): The official constraint context for this Message Assembly.
### 4.2 Internal Schema

A UN/CEFACT internal schema module is a schema instance that contains schema constructs specific to a given root scheme and that therefore is part of a schema set within a specific namespace. The internal schema module will contain schema constructs representing ABIEs that are specific to a given root scheme. These constructs are subject to the same rules as those for reusable ABIEs as provided in sections 4.3.4, 4.3.5, and 4.3.6.

#### 4.2.1 Schema Construct

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

#### 4.2.2 Namespace Scheme

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

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

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

#### 4.2.3 Imports and Includes

The internal schema module does not import or include any other schema module as this is done within the root schema in the same namespace.

### 4.3 Reusable Aggregate Business Information Entities

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

#### 4.3.1 Schema Construct

The reusable ABIE schema will be constructed in a standardized format in order to ensure consistency and ease of use. The specific format is shown below and must adhere to the format of the relevant sections as detailed in Appendix A.
Example 4-4: Structure of Reusable ABIEs Schema Module

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- ====================================================================== -->
<!-- ===== RAM Reusable ABIEs Schema Module                             ===== -->
<!-- ====================================================================== -->
<!-- Module of Reusable ABIEs (Aggregate Business Information Entities),   -->
<!-- Agency:    UN/CEFACT,                                               -->
<!-- Version:   0.3 Rev. 6,                                              -->
<!-- Last change: 25. June 2004                                      -->
<!-- Copyright (C) UN/CEFACT (2004). All Rights Reserved.              -->
... see copyright information ... -->
<xsd:schema
    targetNamespace="... see namespace declaration ..." xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- == Imports == -->
    <!-- == Type Definitions == -->
</xsd:schema>
```

4.3.2 Namespace Scheme

[R 92] The schema module MUST be represented by the token “ram”.

Example 4-5: Namespace of Reusable Aggregate Business Information Entity Schema Module

"urn:un:unece:uncefact:data:draft:ReusableAggregateBusinessInformationEntitySchemaModule:0:3:6"

Example 4-6: Schema-Element of Reusable ABIEs Schema Module

```xml
<xsd:schema
    targetNamespace="... see namespace declaration ..." xmlns:ram="urn:un:unece:uncefact:data:draft:ReusableAggregateBusinessInformationSchemaModule:0:3:6">
```

4.3.3 Imports and Includes

[R 93] The `ram:ReusableAggregateBusinessInformationEntity` schema MUST import the following schema modules:

- `udt:UnqualifiedDataType` Module
- `qdt:QualifiedDataType` Module

Example 4-7: Import of required modules

```xml
<!-- == Imports == -->
<!-- == Import of Qualified Data Type Schema Module (QDT) == -->
<xsd:import
    namespace="... see namespace declaration ..." schemaLocation="QualifiedDataTypeSchemaModule_0.3.6.xsd"/>
```

NamingAndDesignRules_1.0.doc Page 36   8/12/2004
4.3.4 Type Definitions

[R 94] For every object class (ABIE) identified in the UN/CEFACT syntax-neutral model, a named

```
xsd:complexType MUST be defined.
```

[R 95] The name of the ABIE `xsd:complexType` MUST be the `ccts:DictionaryEntryName` with the

```
separators removed and with the "Details" suffix replaced with "Type".
```

For every complex type definition based on an ABIE object class, its `xsd:content model` will be

```
defined such that it reflects each property of the object class as a local element declaration, with its
cardinality and sequencing within the schema `xsd:content model` determined by the details of the
source business information entity (ABIE).
```

[R 96] Every aggregate business information entity (ABIE) `xsd:complexType definition` `xsd:content

```
model MUST use the `xsd:sequence` and/or `xsd:choice` elements with appropriate local
element declarations to reflect each property (BBIE or ASBIE) of its class.
```

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

No complex type may contain a sequence followed by another sequence or a choice followed by another
choice. However, it is permissible to alternate sequence and choice as in example 4-10.

**Example 4-8: Sequence within an object class**

```
<xsd:complexType name="AccountType">
  <xsd:annotation>
    ...see annotation...
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="ID" type="udt:IdentifierType"
      minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        ...see annotation...
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="Status" type="ram:StatusType"
      minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        ...see annotation...
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="Name" type="udt:NameType"
      minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        ...see annotation...
      </xsd:annotation>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

**Example 4-9: Choice**

```
<xsd:complexType name="LocationType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:choice>
    <xsd:element name="GeoCoordinate" type="ram:GeoCoordinateType"
Example 4-10: Sequence + Choice within Object Class "PeriodType"

```xml
<xsd:complexType name="PeriodType">
  ...
  <xsd:sequence>
    <xsd:element name="DurationDateTime" type="qdt:DurationDateTimeType" minOccurs="0" maxOccurs="unbounded">
      ...
    </xsd:element>
    <xsd:choice>
      <xsd:sequence>
        <xsd:element name="StartTime" type="udt:TimeType" minOccurs="0">
          ...
        </xsd:element>
        <xsd:element name="EndTime" type="udt:TimeType" minOccurs="0">
          ...
        </xsd:element>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element name="StartDate" type="udt:DateType" minOccurs="0">
          ...
        </xsd:element>
        <xsd:element name="EndDate" type="udt:DateType" minOccurs="0">
          ...
        </xsd:element>
      </xsd:sequence>
      <xsd:sequence>
        <xsd:element name="StartDateTime" type="udt:DateTimeType" minOccurs="0">
          ...
        </xsd:element>
        <xsd:element name="EndDateTime" type="udt:DateTimeType" minOccurs="0">
          ...
        </xsd:element>
      </xsd:sequence>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

[R 98] The order and cardinality of the elements within an ABIE `xsd:complexType` MUST be according to the structure of the ABIE as defined in the model.

Example 4-11: Type definition of an ABIE

```xml
<!-- ===== Type Definitions                                        =====
<!-- ===================================================================
<xsd:complexType name="AccountType">
  ...
</xsd:complexType>
```
4.3.5 Element Declarations

[R 99] For every attribute of an object class (BBIE) identified in the UN/CEFACT syntax-neutral model, a named \texttt{xsd:element} MUST be locally declared within the \texttt{xsd:complexType} representing the ABIE.

[R 100] Each BBIE element name declaration MUST be based on the property term and qualifiers and the representation term of the basic business information entity (BBIE). If there are duplicate words in the property term and representation terms of the source dictionary entry name, then the duplicate words must be removed.

[R 101] The BBIE element MUST be based on an appropriate data type that is defined in the UN/CEFACT \texttt{qd.t:QualifiedDataType} or \texttt{udt:UnqualifiedDataType} schema modules.

[R 102] For every association (ASBIE) identified in the UN/CEFACT syntax-neutral model, a named \texttt{xsd:element} MUST be locally declared within the \texttt{xsd:complexType} representing the ABIE.

[R 103] Each ASBIE element name declaration MUST be based on the property term and object class of the association business information entity (ASBIE). If there are duplicate words in the property term and representation terms of the source dictionary entry name, then the duplicate words MUST be removed.

[R 104] The element representing an association business information entity (ASBIE) MUST be of the complex type corresponding to its associated aggregate business information (ABIE).

Example 4-12: Element declaration within an ABIE

\begin{verbatim}
... see type definition ...
<xsd:element name="ID" type="udt:IdentifierType"
  minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:element>

<xsd:element name="Status" type="ram:StatusType"
  minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:element>

<xsd:element name="Name" type="udt:NameType"
  minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:element>

<xsd:element name="CurrencyCode" type="qdt:CurrencyCodeType"
  minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:element>
\end{verbatim}
4.3.6 Annotation

[R 105] Each xsd:complexType and xsd:element definition MUST use the xsd:annotation element.

[R 106] A xsd:annotation element declaration MUST appear immediately after the starting tag of the
xsd:complexType or xsd:element.

[R 107] For every ABIE xsd:complexType definition a structured set of annotations MUST be present
in the following pattern:

- UniqueID (mandatory): The identifier that references an Aggregate Business
  Information Entity instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be ABIE.
- DictionaryEntryName (mandatory): The official name of an Aggregate Business Information Entity.
- VersionID (mandatory): An indication of the evolution over time of an Aggregate Business Information Entity instance.
- DefinitionText (mandatory): The semantic meaning of an Aggregate Business Information Entity.
- ObjectClassTermName (mandatory): The Object Class Term of the associated Aggregate Core Component.
- QualifierTermName (optional): Qualifies the Object Class Term of the associated Aggregate Core Component.
- UsageRuleText (optional, repetitive): A constraint that describes specific conditions that are applicable to the Aggregate Business Information Entity.
- BusinessTermName (optional, repetitive): A synonym term under which the Aggregate Business Information Entity is commonly known and used in the business.
- BusinessProcessContext (optional, repetitive): The business process with which this Aggregate Business Information Entity is associated.
- GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this Aggregate Business Information Entity.
- OfficialConstraintContext (optional, repetitive): The official constraint context for this Aggregate Business Information Entity.
- ProductContext (optional, repetitive): The product context for this Aggregate Business Information Entity.
- IndustryContext (optional, repetitive): The industry context for this Aggregate Business Information Entity.
- BusinessProcessRoleContext (optional, repetitive): The role context for this Aggregate Business Information Entity.
Example 4-13: Example of annotation of an ABIE

```xml
<xsd:complexType name="AccountType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UN00000001</ccts:UniqueID>
      <ccts:CategoryCode>ABIE</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Account. Details</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:Definition>
        A business arrangement whereby debits and/or credits arising from transactions are recorded. This could be with a bank, i.e. a financial account, or a trading partner offering supplies or services 'on account', i.e. a commercial account.
      </ccts:Definition>
    </xsd:documentation>
  </xsd:annotation>
</xsd:complexType>
```

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

- **UniqueID (mandatory):** The identifier that references a Basic Business Information Entity instance in a unique and unambiguous way.
- **CategoryCode (mandatory):** The category to which the object belongs. In this case the value will always be BBIE.
- **Dictionary Entry Name (mandatory):** The official name of a Basic Business Information Entity.
- **VersionID (mandatory):** An indication of the evolution over time of a Basic Business Information Entity instance.
- **Definition (mandatory):** The semantic meaning of a Basic Business Information Entity.
- **CardinalityText (mandatory):** Indication whether the Basic Business Information Entity Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Aggregate Business Information Entity.
- **ObjectClassTermName (mandatory):** The Object Class Term of the associated Aggregate Core Component.
- **PropertyTermName (mandatory):** The Property Term of the associated Aggregate Core Component.
- **RepresentationTermName (mandatory):** The Representation Term of the associated Core Component in the associated Aggregate Core Component.
• QualifierTermName (optional): Qualifies the Property Term of the associated Core Component Property in the associated Aggregate Core Component.

• UsageRuleText (optional, repetitive): A constraint that describes specific conditions that are applicable to the Basic Business Information Entity.

• BusinessTermName (optional, repetitive): A synonym term under which the Basic Business Information Entity is commonly known and used in the business.

• BusinessProcessContext (optional, repetitive): The business process with which this Business Information Entity is associated.

• GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this Business Information Entity.

• OfficialConstraintContext (optional, repetitive): The official constraint context for this Business Information Entity.

• ProductContext (optional, repetitive): The product context for this Business Information Entity.

• IndustryContext (optional, repetitive): The industry context for this Business Information Entity.

• BusinessProcessRoleContext (optional, repetitive): The role context for this Business Information Entity.

• SupportingRoleContext (optional, repetitive): The supporting role context for this Business Information Entity.

• SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this Business Information Entity.

• Example (optional, repetitive): Example of a possible value of a Basic Business Information Entity.

**Example 4-14: Annotation of a BBIE**

```
<xsd:element name="ID" type="udt:IdentifierType"
 minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UN00000002</ccts:UniqueID>
      <ccts:CategoryCode>BBIE</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Account. Identifier</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>The identification of a specific account.</ccts:DefinitionText>
      <ccts:CardinalityText>0..n</ccts:CardinalityText>
      <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
      <ccts:PropertyTermName>Identifier</ccts:PropertyTermName>
      <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
      <ccts:BusinessTermName>Account Number</ccts:BusinessTermName>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

[R 109] For every ASBIE `xsd:element` declaration a structured set of annotations MUST be present in the following pattern:

• UniqueID (mandatory): The identifier that references an Association Business Information Entity instance in a unique and unambiguous way.
CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be ASBIE.

DictionaryEntryName (mandatory): The official name of an Association Business Information Entity.

VersionID (mandatory): An indication of the evolution over time of an Association Business Information Entity instance.

DefinitionText (mandatory): The semantic meaning of an Association Business Information Entity.

CardinalityText (mandatory): Indication whether the Association Business Information Entity Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Aggregate Business Information Entity.

ObjectClassTermName (mandatory): The Object Class Term of the associated Aggregate Core Component.

PropertyTermName (mandatory): The Property Term of the associated Aggregate Core Component.

AssociatedObjectClassTermName (mandatory): The Associated Object Class Term of the associated Aggregate Core Component.

QualifierTermName (optional): Qualifies the Property Term of the associated Core Component Property in the associated Aggregate Core Component.

UsageRuleText (optional, repetitive): A constraint that describes specific conditions that are applicable to the Association Business Information Entity.

BusinessTermName (optional, repetitive): A synonym term under which the Association Business Information Entity is commonly known and used in the business.

BusinessProcessContext (optional, repetitive): The business process with which this Association Business Information Entity is associated.

GeopoliticalorRegionContext (optional, repetitive): The geopolitical/region contexts for this Association Business Information Entity.

OfficialConstraintContext (optional, repetitive): The official constraint context for this Association Business Information Entity.

ProductContext (optional, repetitive): The product context for this Association Business Information Entity.

IndustryContext (optional, repetitive): The industry context for this Association Business Information Entity.

BusinessProcessRoleContext (optional, repetitive): The role context for this Association Business Information Entity.

SupportingRoleContext (optional, repetitive): The supporting role context for this Association Business Information Entity.

SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this Association Business Information Entity.
Example (optional, repetitive): Example of a possible value of an Association Business Information Entity.

Example 4-15: Annotation of an ASBIE

```xml
<xsd:element name="Status" type="ram:StatusType"
  minOccurs="0" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UN00000003</ccts:UniqueID>
      <ccts:CategoryCode>ASCC</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Account. Status</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>Associated status information related to account details.</ccts:DefinitionText>
      <ccts:CardinalityText>0..n</ccts:CardinalityText>
      <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
      <ccts:PropertyTermName>Status</ccts:PropertyTermName>
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

4.4 Core Component Type

4.4.1 Use of Core Component Type Module

The purpose of the core component type module is to define the core component types on which the unqualified data types are based. There will only be a core component type defined for the primary representation term. This module is only for reference and will not be included/imported in any schema. The normative formatted schema for the CoreComponentType module is contained in Appendix F.

4.4.2 Schema Construct

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

Example 4-16: Structure of Core Component Type Schema Module

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- ====================================================================== -->
<!-- ===== CCTS Core Component Types Schema Module                    ===== -->
<!-- ====================================================================== -->
<!-- Module of Core Component Types, Agency: UN/CEFACT, OAGi, Version: 0.3 Rev. 6, Last change: 25. June 2004 -->
<!-- Copyright (C) UN/CEFACT (2004). All Rights Reserved. -->
... see copyright information ...

<!-- see type definitions -->
</xsd:schema>
```
## 4.4.3 Namespace Scheme

[R 110] The core component type (CCT) schema module MUST be represented by the token "cct".

### Example 4-17: Namespace of Core Component Type Schema Module

```
"urn:un:unece:uncefact:documentation:draft:CCTS_CCT_SchemaModule:0:3:6"
```

### Example 4-18: Namespace of Core Component Type Schema Module

```xml
<xsd:schema
targetNamespace="urn:un:unece:uncefact:documentation:draft:CCTS_CCT_SchemaModule:0:3:6"
xmlns:cct="urn:un:unece:uncefact:documentation:draft:CCTS_CCT_SchemaModule:0:3:6"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

## 4.4.4 Imports and Includes

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

### 4.4.5 Type Definitions

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

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

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

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

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

### Example 4-19: Type definition of a CCT

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

[R 116] A `xsd:attribute` MUST be declared for each supplementary component.

[R 117] Each supplementary component `xsd:attribute` "name" MUST be the supplementary component name with the separators and spaces removed.

Example 4-10: Attribute of a supplementary component

```xml
  ... see type definition ...
  <xsd:attribute name="currencyID" type="xsd:token" use="optional">
    ... see annotation ...
  </xsd:annotation>
  <xsd:attribute name="currencyCodeListVersionID" type="xsd:token"
    use="optional">
    ... see annotation ...
  </xsd:annotation>
</xsd:attribute>
  ... see type definition ...
```

4.4.7 Extension and Restriction

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

4.4.8 Annotation

[R 118] Each `xsd:complexType` definition MUST use the `xsd:annotation` element.

[R 119] An `xsd:annotation` element declaration MUST appear immediately after the starting tag of the `xsd:complexType`.

[R 120] For every `cct:CoreComponentType xsd:complexType` definition a structured set of annotations MUST be present in the following pattern:

- **UniqueID** (mandatory): The identifier that references the Core Component Type instance in a unique and unambiguous way.
- **CategoryCode** (mandatory): The category to which the object belongs. In this case the value will always be CCT.
- **DictionaryEntryName** (mandatory): The official name of a Core Component Type.
- **VersionID** (mandatory): An indication of the evolution over time of a Core Component Type instance.
- **DefinitionText** (mandatory): The semantic meaning of a Core Component Type.
- **RepresentationTermName** (mandatory): The primary representation term of the Core Component Type.
UsageRuleText (optional, repetitive): A constraint that describes specific conditions that are applicable to the Core Component Type.

BusinessTermName (optional, repetitive): A synonym term under which the Core Component Type is commonly known and used in the business.

Example (optional, repetitive): Example of a possible value of a Core Component Type.

Example 4-11: Annotation of a CCT

```
... see type definition ...
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID>UNDT000001</ccts:UniqueID>
    <ccts:CategoryCode>CCT</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText>A number of monetary units specified in a currency where the unit of the currency is explicit or implied.</ccts:DefinitionText>
    <ccts:RepresentationTermName>Amount</ccts:RepresentationTermName>
  </xsd:documentation>
</xsd:annotation>
... see type definition ...
```

[R 121] For every supplementary component `xsd:attribute` declaration a structured set of annotations MUST be present in the following pattern:

- **UniqueID** (mandatory): The identifier that references a Supplementary Component instance in a unique and unambiguous way.
- **CategoryCode** (mandatory): The category to which the object belongs. In this case the value will always be SC.
- **Dictionary Entry Name** (mandatory): The official name of a Supplementary Component.
- **VersionID** (mandatory): An indication of the evolution over time of a Supplementary Component instance.
- **DefinitionText** (mandatory): The semantic meaning of a Supplementary Component.
- **CardinalityText** (mandatory): Indication whether the Supplementary Component Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Core Component Type.
- **PropertyTermName** (mandatory): The Property Term of the associated Supplementary Component.
- **RepresentationTermName** (mandatory): The Representation term of the associated Supplementary Component.
- **UsageRuleText** (optional, repetitive): A constraint that describes specific conditions that are applicable to the Supplementary Core Component.
- **Example** (optional, repetitive): Example of a possible value of a Basic Core Component.
Example 4-12: Annotation of a supplementary component

```xml
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID>UNDT000001-SC2</ccts:UniqueID>
    <ccts:CategoryCode>SC</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Amount. Currency. Identifier</ccts:DictionaryEntryName>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText>The currency of the amount.</ccts:DefinitionText>
    <ccts:CardinalityText>0..1</ccts:CardinalityText>
    <ccts:PropertyTermName>Currency</ccts:PropertyTermName>
    <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
    <ccts:UsageRuleText>It is recommended that the currency code is provided in this SC rather than in a separately defined BBIE.</ccts:UsageRuleText>
  </xsd:documentation>
</xsd:annotation>
```

4.5 Unqualified Data Type

4.5.1 Use of Unqualified Data Type Module

The unqualified data type schema module will define data types for all primary and secondary representation terms as specified in the CCTS. All data types will be `xsd:complexType` or `xsd:simpleType` and will be without facet restrictions.

4.5.2 Schema Construct

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

Example 4-13: Structure of unqualified data type schema module

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- Module of Unqualified Data Types, 
Agency: UN/CEFACT, 
Version: 0.3 Rev. 6, 
Last change: 25. June 2004 
Copyright (C) UN/CEFACT (2004). All Rights Reserved. 
-->
<xsd:schema targetNamespace=
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- Imports 
  -->
  <!-- Primary RT: Amount. Type 
  -->
  <xsd:complexType name="AmountType">
    <!-- see type definition ... 
    -->
  </xsd:complexType>
</xsd:schema>
```
4.5.3 Namespace Scheme

[R 122] The unqualified data types schema module namespace MUST be represented by the token "udt".

Example 4-14: Namespace of unqualified data types scheme module

```
"urn:un:unece:uncefact:data:draft:UnqualifiedDataTypesSchemaModule:0:3:6"
```

Example 4-15: Schema-element of unqualified data types scheme module

```
<xsd:schema

targetNamespace=
"urn:un:unece:uncefact:data:draft:UnqualifiedDataTypesSchemaModule:0:3:6"
xmins:udt=
"urn:un:unece:uncefact:data:draft:UnqualifiedDataTypesSchemaModule:0:3:6"
xmins:xsd="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified" attributeFormDefault="unqualified">
```

4.5.4 Imports and Includes

[R 123] The `udt:UnqualifiedDataType` schema MUST import the following schema modules:

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

Example 4-16: Imports

```
<!-- ===== Imports                                                 ===== -->
<!-- =================================================================== -->
<!-- ===== Imports of Code Lists                                   ===== -->
<!-- =================================================================== -->
<xsd:import

namespace=
"urn:un:unece:uncefact:codelist:draft:UNECE_Calculation_Amount:6:1"
schemaLocation="http://www.unece.org/uncefact/codelist/UNECE_Calculation_Amount_6.1_draft.xsd"/>
<!-- ===== Imports of Identifier Lists                            ===== -->
<!-- =================================================================== -->
<xsd:import

namespace="urn:un:unece:uncefact:codelist:ISO_Country_Identifier:2:4"
schemaLocation="http://www.unece.org/uncefact/codelist/ISO_Country_Identifier_2.4_draft.xsd"/>
```

4.5.5 Type Definitions

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

[R 124] A `udt:UnqualifiedDataType` MUST be defined for each approved primary and secondary representation terms identified in the CCTS Permissible Representation Terms table.

[R 125] The name of each `udt:UnqualifiedDataType` MUST be the dictionary entry name of the primary or secondary representation term, with "Type" at the end and the separators and spaces removed.

In accordance with rules and principles in this document, the unqualified data types will be based on `xsd:built-in` data types whenever the facets of the `xsd:built-in` data type are equivalent to the supplementary components for that data type.

[R 126] For every `udt:UnqualifiedDataType` whose supplementary components map directly to the properties of a built-in `xsd:DataType`, the `udt:UnqualifiedDataType` MUST be defined as a named `xsd:simpleType` in the `udt:UnqualifiedDataType` schema module.
Every udt:UnqualifiedDataType defined as a xsd:simpleType MUST contain one xsd:restriction element. This xsd:restriction element MUST include an xsd:base attribute that defines the specific built-in xsd:dataType required for the content component. When the unqualified data type does not directly map to an xsd:simpleType due to the supplementary components needing to be expressed, it will be defined as an xsd:complexType.

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

Every udt:UnqualifiedDataType xsd:complexType definition MUST contain one xsd:simpleContent element.

Every udt:UnqualifiedDataType xsd:complexType xsd:simpleContent element MUST contain one xsd:extension element. This xsd:extension element must include an xsd:base attribute that defines the specific xsd:built-in data type required for the content component.

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

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

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

Each CCT code type or CCT identifier type supplementary component that has a restriction identified in CCTS, MUST have the name of the CCT supplementary component with the primary representation term removed at the beginning. The xsd:type for this xsd:attribute must be the name of the code list or identifier scheme containing the restricted set of values. The xsd:attribute must contain an xsd:use attribute who’s value must be set to required.

Example 4-17: Type definitions of unqualified data types

```xml
<xs:complexType name="AmountType">
  <xs:annotation>...
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xsd:decimal">
      <xs:attribute name="currencyID" type="clm54217:CurrencyCodeContentType" use="required"/>
      <xs:annotation>...
      </xs:annotation>
    </xs:extension>
    <xs:attribute name="currencyCodeListVersionID" type="xsd:token" use="optional"/>
    <xs:annotation>...
    </xs:attribute>
  </xs:simpleContent>
</xs:complexType>
```

```xml
<xs:complexType name="BinaryObjectType">
  <xs:annotation>...
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="xsd:base64Binary">
      <xs:attribute name="complexTypeID" type="clm54217:BinaryObjectType" use="required"/>
      <xs:annotation>...
      </xs:annotation>
    </xs:extension>
    <xs:attribute name="binaryObjectID" type="clm54217:BinaryObjectIDType" use="required"/>
    <xs:annotation>...
    </xs:attribute>
  </xs:simpleContent>
</xs:complexType>
```
4.5.6 Attribute Declarations

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

Example 4-18: Attribute declarations of unqualified data types

```xml
<xsd:simpleContent>
  <xsd:restriction base="cct:AmountType">
    <xsd:attribute name="currencyID" type="clm54217:CurrencyCodeContentType" use="required">
      <xsd:annotation>
        ... see annotation ...
      </xsd:annotation>
    </xsd:attribute>
  </xsd:restriction>
</xsd:simpleContent>
```

4.5.7 Restriction

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

4.5.8 Annotation

[R 134] Each `udt:UnqualifiedDataType xsd:complexType` and `xsd:simpleType` definition MUST use the `xsd:annotation` element.

[R 135] An `xsd:annotation` element declaration MUST appear immediately after the starting tag of the `xsd:complexType` or `xsd:simpleType`. 
For every `xsd:complexType` or `xsd:simpleType` definition a structured set of annotations MUST be present in the following pattern:

- **UniqueID** (mandatory): The identifier that references a Core Component Type instance in a unique and unambiguous way.
- **CategoryCode** (mandatory): The category to which the object belongs. In this case the value will always be UDT.
- **DictionaryEntryName** (mandatory): The official name of a Core Component Type.
- **VersionID** (mandatory): An indication of the evolution over time of a Core Component Type instance.
- **DefinitionText** (mandatory): The semantic meaning of a Core Component Type.
- **RepresentationTermName** (mandatory): The primary or secondary representation term of the associated Core Component Type.
- **UsageRuleText** (optional, repetitive): A constraint that describes specific conditions that are applicable to the Core Component Type.
- **BusinessTermName** (optional, repetitive): A synonym term under which the Core Component Type is commonly known and used in the business.
- **Example** (optional, repetitive): Example of a possible value of a Core Component Type.

**Example 4-19: Annotation of unqualified type definition**

```xml
... see complex type definition ...
<xsd:annotation>
<xsd:documentation xml:lang="en">
  <ccts:UniqueID>UNDT000001</ccts:UniqueID>
  <ccts:CategoryCode>UDT</ccts:CategoryCode>
  <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText> A number of monetary units specified in a currency where the unit of the currency is explicit or implied.</ccts:DefinitionText>
  <ccts:RepresentationTermName>Amount</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:annotation>
... see complex type definition ...
```

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

- **UniqueID** (mandatory): The identifier that references a Supplementary Component of a Core Component Type instance in a unique and unambiguous way.
- **CategoryCode** (mandatory): The category to which the object belongs. In this case the value will always be SC.
- **Dictionary Entry Name** (mandatory): The official name of a Supplementary Component.
- **VersionID** (mandatory): An indication of the evolution over time of a Supplementary Component instance.
Example 4-20: Annotation of a supplementary component

```xml
... see complex type definition ...<xsd:attribute name="currencyID" type="iso4217:CurrencyCodeContentType" use="required">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UNDT000001-SC2</ccts:UniqueID>
      <ccts:CategoryCode>SC</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Amount. Currency. Identifier</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>The currency of the amount.</ccts:DefinitionText>
      <ccts:CardinalityText>0..1</ccts:CardinalityText>
      <ccts:PropertyTermName>Currency</ccts:PropertyTermName>
      <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
      <ccts:UsageRuleText>It is recommended that the currency code is provided in this SC rather than in a separately defined BBIE.</ccts:UsageRuleText>
    </xsd:documentation>
  </xsd:annotation>
</xsd:attribute>
... see complex type definition ...
```

4.6 Qualified Data Type

The data types defined in the unqualified data types schema module are `xsd:complexType` or `xsd:simpleTypes` without facet restrictions. These types are intended to be suitable as the `xsd:base` type for some but not all expected complex types. As business process modeling reveals the need for specialized data types, new ‘qualified’ types will need to be defined. These new ‘qualified’ data types must be restricted technically by facets and semantically by one or more additional qualifier terms. These ‘qualified’ data types will be defined using `xsd:restriction` on the unqualified data types or can be based on new specific simple types.

4.6.1 Use of Qualified Data Type Module

To ensure consistency of qualified data types with the UN/CEFACT modularity and reuse goals will require creating a single schema module that defines all qualified data types. The qualified data type schema module name must follow the UN/CEFACT module naming approach. The qualified data type schema module will be used by the resuable ABIE schema module and all root schema modules.
4.6.2 Schema Construct

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

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by Gunther Stuhec (UBL) -->
<!-- ============================================================== -->
<!-- ===== QDT Qualified Data Type Schema Module                     ===== -->
<!-- ============================================================== -->
<!-- Module of Qualified Data Types,                                -->
<!-- Agency: UN/CEFACT, OAGi,                                        -->
<!-- Version: 0.3 Rev. 6,                                            -->
<!-- Last change: 25. June 2004                                     -->
<!-- Copyright (C) UN/CEFACT (2004). All Rights Reserved.            -->
...
</xsd:schema>
```

4.6.3 Namespace Scheme

[R 138] The UN/CEFACT:QualifiedDataType schema module namespace MUST be represented by the token “qdt”.

Example 4-21: Namespace name

```xml
"urn:un:unece:uncefact:data:draft:QualifiedDataTypeSchemaModule:0:3:6"
```

Example 4-21: Schema element

```xml
<xsd:schema targetNamespace="urn:un:unece:uncefact:data:draft:
QualifiedDataTypeSchemaModule:0:3:6"
xmlns:udt="urn:un:unece:uncefact:data:draft:
QualifiedDataTypeSchemaModule:0:3:6"
xmlns:qdt="urn:un:unece:uncefact:data:draft:
QualifiedDataTypeSchemaModule:0:3:6"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

4.6.4 Imports and Includes

Qualified data types will be derived from data types defined in the unqualified data types, code list, and identifier list schema modules. These schema modules reside in their own namespaces and as such will be imported rather than included in the qualified data types schema module.
The `<qdt:QualifiedDataType>` schema module MUST import the:

- `<udt:UnqualifiedDataType>` schema module
- `<ids:IdentifierList>` schema modules
- `<clm:CodeList>` schema module

Note: If needed, relevant UN/CEFACT and external code list and identifier scheme schema modules may be imported.

```xml
<xsd:import namespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:0:3:6" 
schemaLocation="http://www.unece.org/uncefact/data/UnqualifiedDataTypeSchemaModule_0.3.6_draft.xsd"/>
```

### 4.6.5 Type Definitions

[R 140] Where required to change facets of an existing `<udt:UnqualifiedDataType>`, a new data type MUST be defined in the `<qdt:QualifiedDataType>` schema module.

[R 141] If a `<qdt:QualifiedDataType>` is based on an `<udt:UnqualifiedDataType>` that is an `<xsd:simpleType>`, the `<qdt:QualifiedDataType>` MUST also be an `<xsd:simpleType>.

[R 142] If a `<qdt:QualifiedDataType>` is based on an `<udt:UnqualifiedDataType>` that is a `<xsd:complexType>`, the `<qdt:QualifiedDataType>` MUST also be an `<xsd:complexType>.

[R 143] If a `<qdt:QualifiedDataType>`’s restrictions map directly to the properties of a `<xsd:builtin>` data type, the `<qdt:QualifiedDataType>` MUST be defined as a named `<xsd:simpleType>` in the `<qdt:QualifiedDataType>` schema module.

[R 144] The name of a `<qdt:QualifiedDataType>` MUST be the dictionary entry name of its base `<udt:UnqualifiedDataType>` with an appropriate qualifier or qualifiers.

[R 145] Each `<qdt:QualifiedDataType>` `<xsd:complexType>` definition MUST contain one `<xsd:simpleContent>` element.

[R 146] The `<qdt:QualifiedDataType>` `<xsd:complexType>` definition `<xsd:simpleContent>` element MUST contain one `<xsd:restriction>` element with an `<xsd:base>` attribute whose value is equal to an `<udt:UnqualifiedDataType>` or an externally imported code list.

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

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

4.6.7 Extension and Restriction

[R 147] The <qdt:QualifiedDataType> xsd:complexType definition xsd:simpleContent element MUST only restrict attributes declared in its base type, or must only restrict facets equivalent to allowed supplementary components.

Example 4-22: Qualified Data Type Restricting an Identification Scheme
### 4.6.8 Annotation

- **[R 148]** Each `qdt:QualifiedDataType xsd:complexType and xsd:simpleType` definition MUST use the `xsd:annotation` element.

- **[R 149]** An `xsd:annotation` element declaration MUST appear immediately after the starting tag of the `xsd:complexType or xsd:simpleType`.

- **[R 150]** Every `qdt:QualifiedDataType` definition MUST contain a structured set of annotations in the following sequence and pattern:

  - **UniqueId** (mandatory): The identifier that references a Qualified Data Type instance in a unique and unambiguous way.
  - **CategoryCode** (mandatory): The category to which the object belongs. In this case the value will always be QDT.
  - **DictionaryEntryName** (mandatory): The official name of a Qualified Data Type.
  - **VersionID** (mandatory): An indication of the evolution over time of a Qualified Data Type instance.
  - **DefinitionText** (mandatory): The semantic meaning of a Qualified Data Type.
  - **RepresentationTermName** (mandatory): The Representation Term of the associated Unqualified Data Type.
  - **QualifierTerm** (mandatory): Qualifies the Representation Term in order to differentiate it from its underlying Core Component Type and other Data Types.
  - **UsageRuleText** (optional, repetitive): A constraint that describes specific conditions that are applicable to the Qualified Data Type.
  - **BusinessTermName** (optional, repetitive): A synonym term under which the Qualified Data Type is commonly known and used in the business.
  - **BusinessProcessContext** (optional, repetitive): The business process with which this Qualified Data Type is associated.
  - **GeopoliticalorRegionContext** (optional, repetitive): The geopolitical/region contexts for this Qualified Data Type.
  - **OfficialConstraintContext** (optional, repetitive): The official constraint context for this Qualified Data Type.
  - **ProductContext** (optional, repetitive): The product context for this Qualified Data Type.
  - **IndustryContext** (optional, repetitive): The industry context for this Qualified Data Type.
  - **BusinessProcessRoleContext** (optional, repetitive): The role context for this Qualified Data Type.
SupportingRoleContext (optional, repetitive): The supporting role context for this Qualified Data Type.

SystemCapabilitiesContext (optional, repetitive): The system capabilities context for this Qualified Data Type.

Example (optional, repetitive): Example of a possible value of a Qualified Data Type.

Example 4-23: Annotation of qualified data types

```xml
... see type definition ...
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID/>
    <ccts:CategoryCode>QDT</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Account_Type_Code. Type</ccts:DictionaryEntryName>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText> This code represents the type of an account. </ccts:DefinitionText>
    <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
    <ccts:ObjectClassQualifier>Account</ccts:ObjectClassQualifier>
    <ccts:ObjectClassQualifier>Type</ccts:ObjectClassQualifier>
  </xsd:documentation>
</xsd:annotation>
... see type definition ...
```

[R 151] For every supplementary component `xsd:attribute` declaration a structured set of annotations MUST be present in the following pattern:

- UniqueID (mandatory): The identifier that references a Supplementary Component of a Core Component Type instance in a unique and unambiguous way.
- CategoryCode (mandatory): The category to which the object belongs. In this case the value will always be QDT.
- Dictionary Entry Name (mandatory): The official name of a Supplementary Component.
- VersionID (mandatory): An indication of the evolution over time of a Supplementary Component instance.
- DefinitionText (mandatory): The semantic meaning of a Supplementary Component.
- CardinalityText (mandatory): Indication whether the Supplementary Component Property represents a not-applicable, optional, mandatory and/or repetitive characteristic of the Core Component Type.
- PropertyTermName: The Property Term of the associated Supplementary Component.
- RepresentationTermName: The Representation Term of the associated Supplementary Component.
- UsageRuleText (optional, repetitive): A constraint that describes specific conditions that are applicable to the Supplementary Component.
- BusinessProcessContext (optional, repetitive): The business process with which this Supplementary Component is associated.
4.7 Code Lists

[R 152] Internal code lists MUST NOT duplicate existing external code lists where they are available in a
schema module form that is capable of being imported. External code lists must be used when they exist in schema module form and when they can be directly imported into a schema module. UN/CEFACT may design and use an internal code list where an existing external code list needs to be extended, or where no suitable external code list exists. If a code list is created, the lists should be globally scoped and designed for reuse and sharing.

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

4.7.1 Schema Construct

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

Example 4-24: Structure of code lists

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- ====================================================================== -->
<!-- =====  Code List: Account Type Code ; UNECE === -->
<!-- ====================================================================== -->
<!-- Copyright (C) UN/CEFACT (2004). All Rights Reserved. -->
... see copyright information ...
-->
```
4.7.2 Namespace Scheme

In order to allow for uniqueness with a reasonably short token it is recommended that the token representing the namespace for code lists is constructed based on the identifier of the agency maintaining the code list and the identifier for the specific code list as issued by the maintenance agency. In cases where there is no identifier, the name for the agency and/or code list should be used instead. This will typically be true when proprietary code lists are used.

[R 154] Each UN/CEFACT maintained code list schema module MUST be represented by a unique token constructed as follows:

\[\text{clm[Qualified data type name]}\text{|Code List. Agency. Identifier|Code List. Agency Name. Text}\]

with any repeated words eliminated.

The qualified data type name is required whenever the code list is used for a qualified data type with a restricted set of valid code values.

Example 4-25: Code list tokens

\begin{verbatim}
Code list token for Name type. Code is clm63403
where
6 = the value for UN/ECE in UN/EDIFACT data element 3055 representing the Code List. Agency. Identifier
3403 = UN/EDIFACT data element tag for Name status code representing the Code List. Identification. Identifier

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

Code list token for a proprietary code list for Document Security is clmzzzDocumentSecurity
where
zzz = the value for Mutually defined in UN/EDIFACT data element 3055 representing the Code List. Agency. Identifier

Code Security = the value for Code List. Name. Text
\end{verbatim}

Example 4-26: Schema element of code lists

\begin{verbatim}
<xsd:schema targetNamespace="urn:un:unece:uncefact:codelist:draft:UNECE_Account_Type_Code:D.01C"
xmlns:clm64437="urn:un:unece:uncefact:codelist:draft:UNECE_Account_Type_Code:D.01C"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified" attributeFormDefault="unqualified">

Note: External developers are encouraged to follow the above construct rule when customizing schemas for codelists to ensure that there is no namespace conflict.
4.7.3 Imports and Includes

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

[R 155] Code List schema modules MUST not import or include any other schema modules.

4.7.4 Type Definitions

[R 156] For the top-level element a corresponding top-level xsd:complexType MUST be defined.

[R 157] The name for the xsd:complexType MUST be based on the value of the code list. name. text. with the word “type” appended.

[R 158] Each xsd:complexType MUST contain one xsd:simpleContent element.

[R 159] The xsd:complexType definition xsd:simpleContent element MUST contain one xsd:extension element. This xsd:extension element MUST include an xsd based attribute that defines the specific list containing the actual code values.

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

[R 160] Within each code list module one, and only one, named xsd:simpleType MUST be defined for the content component.

[R 161] The name of the xsd:simpleType MUST be the name of root element based on the value of the code list. name. text. with the word “ContentType” appended.

Example 4-27: Simple type definition of code lists

```xml
<!-- ===== Type Definitions -->
<!-- Code List Type Definition: Account Type Code -->
<xsd:simpleType name="AccountTypeCodeContentType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="2">
      ... see enumeration ...
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
```

[R 162] The xsd:restriction element base attribute value MUST be set to "xsd:token".

[R 163] Each code in the code list MUST be expressed as an xsd:enumeration, where the xsd:value for the enumeration is the actual code value.

Example 4-28: Enumeration facet of code lists

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

NamingAndDesignRules_1.0.doc Page 61  8/12/2004
The XSD facet feature MUST NOT be used in the code list schema module.

### 4.7.5 Element Declarations

- [R 165] For each code list a single root element MUST be globally declared.
- [R 166] The name of root element MUST be based on the code list name, text following the naming rules as defined in section 2.3.
- [R 167] The root element MUST be of a type representing the actual list of code values.

#### Example 4-29: Root element declaration of code lists

```xml
<xs:element name="AccountTypeCode" type="clm64437:AccountTypeCodeContentType"/>
```

### 4.7.6 Extension and Restriction

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

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

Both of these can easily be accommodated in this syntax solution, although they are not likely to be frequently used.

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

#### Example 4-30: Usage of only one Code List

```xml
<xsd:simpleType name="TemperatureMeasureUnitCodeType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:restriction base="clm66411:UnitCodeContentType">
    <xsd:length value="3"/>
    <xsd:enumeration value="BTU">
      <xsd:annotation>
        <xsd:documentation source="code" xml:lang="en">
          <ccts:CodeName>British thermal unit</ccts:CodeName>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="CEL">
      <xsd:annotation>
        <xsd:documentation source="code" xml:lang="en">
          <ccts:CodeName>degree Celsius</ccts:CodeName>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
    <xsd:enumeration value="FAH">
      <xsd:annotation>
        <xsd:documentation source="code" xml:lang="en">
          <ccts:CodeName>degree Fahrenheit</ccts:CodeName>
        </xsd:documentation>
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
```
Example 4-31 Usage of alternative Code Lists

```xml
 xsd:complexType name="PersonPropertyCodeType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:choice>
    <xsd:element ref="clm63479:MaritalCode"/>
    <xsd:element ref="clm63499:GenderCode"/>
  </xsd:choice>
</xsd:complexType>
```

Example 4-32: Combination of Code Lists

```xml
 xsd:simpleType name="AccountDutyCodeType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
  <xsd:union memberTypes="clm64437:AccountTypeCodeContentType
  clm65153:DutyTaxFeeTypeCodeContentType"/>
</xsd:simpleType>
```

4.7.7 Annotation

[R 168] Each `xsd:enumeration` MUST include an annotation documentation providing the code name and the code description.

Example 4-33: Annotation of codes

```xml
 ... 
  <xsd:enumeration value="2">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        <ccts:CodeName>Budgetary account</ccts:CodeName>
        <ccts:CodeDescription>Code identifying a budgetary account.</ccts:CodeDescription>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:enumeration>
... 
```

4.8 Identifier Schemas

When required separate schema modules will be defined for identification schemes that have a token, and optionally a description, and that have the same functionality as a code list. Other identifier schemes should be defined as a qualified or unqualified data type as appropriate.

[R 169] Where an identification scheme is identified (contains an identifier, a token for the identifier, and an optionally, a description) to have the same functionality as a code list and a business requirement exists for enumerating the identifier list, a schema module of the same form as the code list schema module MUST be created.

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

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

The identifier list schema module will follow the general pattern for all UN/CEFACT schema modules.

Following the generic module information, the body of the schema will consist of identifier list definitions of the following general form:

Example 4-34: Structure of identifier lists

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!--  ISO Country Identifier – Identifier List Schema Module -->
<!--  Copyright (C) UN/CEFACT (2004). All Rights Reserved. -->
... see root element declaration ...
... see type definition ...
</xsd:root>
</xsd:schema>
```

4.8.2 Namespace Scheme

[R 172] Each UN/CEFACT maintained identifier list schema module MUST be represented by a unique token constructed as follows:

token = ids[Qualified data type name]<Identification Scheme Agency. Identifier> <Identification Scheme. Identifier>

Example 4-35: Identifier list token

Token for the ISO Country Codes would be: ids53166-1

where:
5 = the Identification Scheme Agency Identifier for ISO in codelist 3055
3166-1 = the Identification Scheme Identifier as allocated by ISO.

Example 4-36: Namespace of Identifier list

"urn:un:unece:uncefact:codelist:draft:ISO_Country_Identifier:2:4"

Example 4-37: Schema element of identifier list

```xml
<xsd:schema
  targetNamespace="urn:un:unece:uncefact:codelist:draft:ISO_Country_Identifier:2:4"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
```
Note: External developers are encouraged to follow the above construct rule when customizing schemas for identifier lists to ensure that there is no namespace conflict.

### 4.8.3 Imports and Includes

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

[R 173] Identifier list schema modules MUST not import or include any other schema modules.

### 4.8.4 Type Definitions

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

[R 174] Within each identifier list module one, and only one, named `xsd:simpleType` MUST be defined for the content component.

[R 175] The name of the `xsd:simpleType` MUST be the name of root element with the word “ContentType” appended.

**Example 4-38: Simple type definition of an identifier list**

```xml
<!-- ===== Type Definitions ===== -->
<!-- ============================================================== -->
<xsd:simpleType name="CountryIdentifierContentType">
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="AU">
      ... see enumeration ...
    </xsd:enumeration>
    <xsd:enumeration value="US">
      <xsd:annotation>
        ... see annotation
      </xsd:annotation>
    </xsd:enumeration>
  </xsd:restriction>
</xsd:simpleType>
```

[R 176] The `xsd:restriction` element base attribute value MUST be set to "xsd:token".

[R 177] Each identifier in the identifier list MUST be expressed as an `xsd:enumeration`, where the `xsd:value` for the enumeration is the actual identifier value.

**Example 4-39: Enumeration facet of an identifier list**

```xml
... see type definition ...
<xsd:enumeration value="AU">
  <xsd:annotation>
    ... see annotation
  </xsd:annotation>
</xsd:enumeration>
<xsd:enumeration value="US">
  <xsd:annotation>
    ... see annotation
  </xsd:annotation>
</xsd:enumeration>
...```

[R 178] The XSD facet feature MUST not be used in the identifier list schema module.

### 4.8.5 Attribute and Element Declarations

[R 179] For each identifier list a single root element MUST be globally declared.
The name of the root element MUST be based on the identification scheme.

The root element MUST be of a type representing the actual list of identifier values.

**Example 4-40: Root element declaration of identifier lists**

```xml
<!-- ===== Root Element ===== -->
<!-- =================================================================== -->
<xsd:element name="CountryIdentifier" type="ids53166:CountryIdentifierContentType"/>
```

### 4.8.6 Extension and Restriction

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

Representation of qualified data type of identifier lists could be
- a combination of several individual identifier lists using `xsd:union`
- a choice between several identifier lists, using `xsd:choice`

Both of these can easily be accommodated in this syntax solution, although they are not likely to be frequently used.

**Example 4-41: Enumeration facet of identifier scheme**

```xml
... see type definition ...
<xsd:enumeration value="AD">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:enumeration>
<xsd:enumeration value="AE">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:enumeration>
<xsd:enumeration value="AF">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
</xsd:enumeration>
... see type definition ...
```

**Example 4-42: Usage of only one identifier scheme**

```xml
<xsd:simpleType name="CountryIdentifierType">
  <xsd:annotation>
    ... see annotation ...
  </xsd:annotation>
<xsd:restriction base="ids53166:CountryIdentifierContentType"/>
</xsd:simpleType>
```

**Example 4-43: Usage of alternative identifier schemes**

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

[R 182] Each \texttt{xsd:enumeration} MUST include an annotation documentation providing the identifier name and optionally the description of the identifier.

Example 4-44: Annotation of Identifiers

```
... 
<xsd:enumeration value="AU">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccd:IdentifierName>Australia</ccd:IdentifierName>
    </xsd:documentation>
  </xsd:annotation>
</xsd:enumeration>
... 
```
5 XML Instance Documents

All UN/CEFACT compliant XML instance documents will be based on UN/CEFACT compliant XML schemas for core components. The instance documents must be valid against the relevant UN/CEFACT compliant XML schemas. The XML instance documents should be readable and understandable for human and application, and should enable reasonably intuitive interactions. It should represent all truncated tag names as described in section 4. A Xpath navigation path should describe the complete semantic understanding by concatenating the nested elements. This navigation path should also reflect the meaning of each dictionary entry name of a BBIE or ASBIE.

5.1 Character Encoding

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

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

5.2 Empty Content

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

UN/CEFACT conformant instance documents MUST NOT contain an element devoid of content.

5.3 xsi:type

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

The xsi:type attribute MUST NOT be used.
Appendix A. Overall Structure

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

1. XML Declaration
2. Schema Module Identification and Copyright Information
3. Schema Start-Tag
4. Includes
5. Imports
6. Root element
7. Type Definitions

1. XML Declaration

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

Example 1-1: XML Declaration

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

2. Schema Module Identification and Copyright Information

Example 2-1: Copyright Information

```xml
<!-- ================================================================== -->
<!-- ===== Examples Schema Module; 0.3 Rev.6                     ===== -->
<!-- ================================================================== -->
<!--
Module: Example
Agency: UN/CEFACT
Version: 0.3 Rev. 6
Last change: 25. June 2004
-->
Copyright (C) UN/CEFACT (2004). All Rights Reserved.
```

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

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

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

-->
3. Schema Start-Tag

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

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

Example 3-1: XML Schema Start Tag

```xml
<xsd:schema
targetNamespace="urn:un:unece:uncefact:data:draft:ExamplesSchemaModule:0:3:6"
xmlns:rsm="urn:un:unece:uncefact:data:draft:ExamplesSchemaModule:0:3:6"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ram="urn:un:unece:uncefact:data:draft:ReusableAggregateBusinessInformationEntitySchemaModule:0:3:6"
xmlns:udt="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:0:3:6"
xmlns:qdt="urn:un:unece:uncefact:data:draft:QualifiedDataTypeSchemaModule:0:3:6"
xmlns:clm65153="urn:un:unece:uncefact:codelist:draft:UNECE_Duty_Or_Tax_Or_Fee_Type_Code:D.01C"
xmlns:clm64405="urn:un:unece:uncefact:codelist:draft:UNECE_Status_Code:D.01C"
xmlns:clm69143="urn:un:unece:uncefact:codelist:draft:UNECE_Relationship_Code:D.01C"
xmlns:clm64437="urn:un:unece:uncefact:codelist:draft:UNECE_Account_Type_Code:D.01C"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
```
4. Includes

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

- Inclusion of the internal ABIE schema module if used

Example 4-1: Includes

```xml
<!-- ================================================================== -->
<!-- =====  Include                                               ===== -->
<!-- ================================================================== -->
<!-- =====  Inclusion of internal ABIE                            ===== -->
<!-- ================================================================== -->
<xsd:include
    namespace="urn:un:unece:uncefact:data:draft:InternalAggregateBusinessInformationEntity
    SchemaModule:0:3:6"
    schemaLocation="http://www.unece.org/uncefact/data/InternalAggregateBusinessInformationEntity
    nEntitySchemaModule_0.3.6_draft.xsd"/>
```
5. Imports

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

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

Example 5-1: Imports

```xml
<xsd:import namespace="urn:un:unece:uncefact:data:draft:ReusableAggregateBusinessInformationEntitySchemaModule:0:3:6" schemaLocation="http://www.unece.org/uncefact/data/ReusableAggregateBusinessInformationEntitySchemaModule_0.3.6_draft.xsd"/>

<xsd:import namespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypeSchemaModule:0:3:6" schemaLocation="http://www.unece.org/uncefact/data/UnqualifiedDataTypeSchemaModule_0.3.6_draft.xsd"/>

<xsd:import namespace="urn:un:unece:uncefact:data:draft:QualifiedDataTypeSchemaModule:0:3:6" schemaLocation="http://www.unece.org/uncefact/data/QualifiedDataTypeSchemaModule_0.3.6_draft.xsd"/>

<xsd:import namespace="urn:un:unece:uncefact:codelist:draft:Account_Type_Code:D.01C" schemaLocation="http://www.unece.org/uncefact/codelist/UNECE_AccountTypeCode_0.3.6_draft.xsd"/>


```

6. Root element

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

Example 6-1:

```xml
<xs:element name="PurchaseOrder" type="exp:PurchaseOrderType">
  <xs:annotation>
    <ccts:UniqueID>UNM0000001</ccts:UniqueID>
    <ccts:CategoryCode>RSM</ccts:CategoryCode>
    <ccts:Name>PurchaseOrder</ccts:Name>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:Description>A document that contains information directly relating to the economic event of ordering products.</ccts:Description>
    <ccts:BusinessDomain>TBG1</ccts:BusinessDomain>
  </xs:annotation>
</xs:element>
```
7. Type Definitions

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

Example 7-1:

```xml
<xs:complexType name="AccountType">
  <xs:annotation>
    <xs:documentation xml:lang="en">
      <ccts:UniqueID>UN00000001</ccts:UniqueID>
      <ccts:CategoryCode>ABIE</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Account. Details</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>A business arrangement whereby debits and/or credits arising from transactions are recorded. This could be with a bank, i.e. a financial account, or a trading partner offering supplies or services 'on account', i.e. a commercial account.</ccts:DefinitionText>
    </xs:documentation>
    <xs:objectClassTermName>Account</xs:objectClassTermName>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="ID" type="udt:IdentifierType" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation xml:lang="en">
          <ccts:UniqueID>UN00000002</ccts:UniqueID>
          <ccts:CategoryCode>BBIE</ccts:CategoryCode>
          <ccts:DictionaryEntryName>Account. Identifier</ccts:DictionaryEntryName>
          <ccts:VersionID>1.0</ccts:VersionID>
          <ccts:DefinitionText>The identification of a specific account.</ccts:DefinitionText>
          <ccts:CardinalityText>0..n</ccts:CardinalityText>
          <xs:objectClassTermName>Account</xs:objectClassTermName>
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Status" type="ram:StatusType" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation xml:lang="en">
          <ccts:UniqueID>UN00000003</ccts:UniqueID>
          <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
          <ccts:DictionaryEntryName>Account. Status</ccts:DictionaryEntryName>
          <ccts:VersionID>1.0</ccts:VersionID>
          <ccts:DefinitionText>Status information related to account details.</ccts:DefinitionText>
          <xs:objectClassTermName>Account</xs:objectClassTermName>
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="Name" type="udt:NameType" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation xml:lang="en">
          <ccts:UniqueID>UN00000004</ccts:UniqueID>
          <ccts:CategoryCode>BBIE</ccts:CategoryCode>
          <ccts:DictionaryEntryName>Account. Name. Text</ccts:DictionaryEntryName>
          <ccts:VersionID>1.0</ccts:VersionID>
          <ccts:DefinitionText>The text name for a specific account.</ccts:DefinitionText>
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```
<ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
<ccts:PropertyTermName>Name</ccts:PropertyTermName>
<ccts:RepresentationTermName>Text</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:element>
<xsd:element name="CurrencyCode" type="qdt:CurrencyCodeType" minOccurs="0" maxOccurs="unbounded">
<xsd:documentation xml:lang="en">
  <ccts:UniqueID>UN00000005</ccts:UniqueID>
  <ccts:CategoryCode>BBIE</ccts:CategoryCode>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText>A code specifying the currency in which monies are held within the account.</ccts:DefinitionText>
  <ccts:CardinalityText>0..n</ccts:CardinalityText>
  <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
  <ccts:PropertyTermName>Currency</ccts:PropertyTermName>
  <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:element>

<xsd:element name="TypeCode" type="qdt:AccountTypeCodeType" minOccurs="0" maxOccurs="unbounded">
<xsd:documentation xml:lang="en">
  <ccts:UniqueID>UN00000006</ccts:UniqueID>
  <ccts:CategoryCode>BBIE</ccts:CategoryCode>
  <ccts:DictionaryEntryName>Account. Type. Code</ccts:DictionaryEntryName>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText>This provides the ability to indicate what type of account this is (checking, savings, etc).<ccts:DefinitionText>
  <ccts:CardinalityText>0..1</ccts:CardinalityText>
  <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
  <ccts:PropertyTermName>Type</ccts:PropertyTermName>
  <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:element>

<xsd:element name="Country" type="ram:CountryType" minOccurs="0" maxOccurs="unbounded">
<xsd:documentation xml:lang="en">
  <ccts:UniqueID>UN00000007</ccts:UniqueID>
  <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
  <ccts:DictionaryEntryName>Account. Country</ccts:DictionaryEntryName>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText>Country information related to account details.</ccts:DefinitionText>
  <ccts:CardinalityText>0..n</ccts:CardinalityText>
  <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
  <ccts:PropertyTermName>Country</ccts:PropertyTermName>
  <ccts:AssociatedObjectClassTermName>Country</ccts:AssociatedObjectClassTermName>
</xsd:documentation>
</xsd:element>

<xsd:element name="Person" type="ram:PersonType" minOccurs="0" maxOccurs="unbounded">
<xsd:documentation xml:lang="en">
  <ccts:UniqueID>UN00000008</ccts:UniqueID>
  <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
  <ccts:DictionaryEntryName>Account. Person</ccts:DictionaryEntryName>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText>Associated person information related to account details. This can be used to identify multiple people related to an account, for instance, the account holder.</ccts:DefinitionText>
  <ccts:CardinalityText>0..n</ccts:CardinalityText>
  <ccts:ObjectClassTermName>Account</ccts:ObjectClassTermName>
  <ccts:PropertyTermName>Person</ccts:PropertyTermName>
  <ccts:AssociatedObjectClassTermName>Person</ccts:AssociatedObjectClassTermName>
</xsd:documentation>
</xsd:element>
Example 7-2: Complete Structure

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- ================================================================== -->
<!-- =====  [MODULENAME] Schema Module; [VERSION] ===== -->
<!-- ================================================================== -->
<xs:schema
  targetNamespace="urn:un:unece:uncefact:data:draft:[MODULENAME]:[VERSION]
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  ... FURTHER NAMESPACES ...
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <!-- =====  Include ===== -->
  <xs:include
    namespace="..."
schemaLocation="..."/>
  <!-- =====  Imports ===== -->
  <xs:import
    namespace="...
    schemaLocation="..."/>
  <!-- =====  Root element ===== -->
</xs:schema>
```

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to UN/CEFACT, except as needed for the purpose of developing UN/CEFACT specifications, in which case the procedures for copyrights defined in the UN/CEFACT 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 UN/CEFACT or its successors or assigns.

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

---

<table>
<thead>
<tr>
<th>ccts:UniqueID</th>
<th>UN00000009</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccts:CategoryCode</td>
<td>ASBIE</td>
</tr>
<tr>
<td>ccts:DictionaryEntryName</td>
<td>Account. Organisation</td>
</tr>
</tbody>
</table>
<xsd:element name="[ELEMENTNAME]" type="[TOKEN]:[TYPENAME]">
</xsd:element>

<!-- Type Definitions -->

<xs:schema>

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

</xs:schema>
Appendix B. Annotation Templates

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

<!-- Root Schema Documentation -->
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID></ccts:UniqueID>
    <ccts:CategoryCode>RSM</ccts:CategoryCode>
    <ccts:Name></ccts:Name>
    <ccts:VersionID></ccts:VersionID>
    <ccts:Description></ccts:Description>
    <ccts:BusinessDomain></ccts:BusinessDomain>
    <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
    <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
    <ccts:ProductContext></ccts:ProductContext>
    <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
    <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
    <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
    <ccts:Example></ccts:Example>
  </xsd:documentation>
</xsd:annotation>

<!-- ABIE's Documentation -->
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID></ccts:UniqueID>
    <ccts:CategoryCode>ABIE</ccts:CategoryCode>
    <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
    <ccts:VersionID></ccts:VersionID>
    <ccts:DefinitionText></ccts:DefinitionText>
    <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
    <ccts:QualifierTermName></ccts:QualifierTermName>
    <ccts:UsageRuleText></ccts:UsageRuleText>
    <ccts:BusinessTermName></ccts:BusinessTermName>
    <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
    <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
    <ccts:ProductContext></ccts:ProductContext>
    <ccts:BusinessProcessRoleContext></ccts:BusinessProcessRoleContext>
    <ccts:SupportingRoleContext></ccts:SupportingRoleContext>
    <ccts:SystemCapabilitiesContext></ccts:SystemCapabilitiesContext>
    <ccts:Example></ccts:Example>
  </xsd:documentation>
</xsd:annotation>

<!-- BBIE's Documentation -->
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    <ccts:UniqueID></ccts:UniqueID>
    <ccts:CategoryCode>BBIE</ccts:CategoryCode>
    <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
    <ccts:VersionID></ccts:VersionID>
    <ccts:DefinitionText></ccts:DefinitionText>
    <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
  </xsd:documentation>
</xsd:annotation>

NamingAndDesignRules_1.0.doc Page 78   8/12/2004
<ccts:PropertyTermName/>
</ccts:PropertyTermName>
</ccts:CategoryCode>
</ccts:DictionaryEntryName>
</ccts:VersionID>
</ccts:DefinitionText>
</ccts:CardinalityText>
</ccts:ObjectClassTermName>
</ccts:PropertyTermName>
</ccts:QualifierTermName>
</ccts:UsageRuleText>
</ccts:BusinessTermName>
</ccts:BusinessProcessContext>
</ccts:GeopoliticalOrRegionContext>
</ccts:OfficialConstraintContext>
</ccts:BusinessTermName>
</ccts:UsageRuleText>
</ccts:QualifierTermName>
</ccts:AssociatedObjectClassTermName>
</ccts:ObjectClassTermName>
</ccts:CardinalityText>
</ccts:DefinitionText>
</ccts:VersionID>
</ccts:DictionaryEntryName>
</ccts:CategoryCode>
</ccts:UniqueID>
</ccts:Documentation>
</ccts:IndustryContext>
</ccts:CardinalityText>
</ccts:DefinitionText>
</ccts:VersionID>
</ccts:DictionaryEntryName>
</ccts:CategoryCode>
</ccts:UniqueID>
</ccts:Documentation>
<!-- Qualified Data Types Documentation -->
<xs:annotation xml:lang="en">
  <ccts:CategoryCode>QDT</ccts:CategoryCode>
  <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
  <ccts:VersionID></ccts:VersionID>
  <ccts:DefinitionText></ccts:DefinitionText>
  <ccts:RepresentationTermName></ccts:RepresentationTermName>
  <ccts:QualifierTermName></ccts:QualifierTermName>
  <ccts:UsageRuleText></ccts:UsageRuleText>
  <ccts:BusinessTermName></ccts:BusinessTermName>
  <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
  <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
  <ccts:ProductContext></ccts:ProductContext>
</xs:annotation>

<xs:documentation xml:lang="en">
  <ccts:UniqueID></ccts:UniqueID>
  <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
  <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
  <ccts:VersionID></ccts:VersionID>
  <ccts:DefinitionText></ccts:DefinitionText>
  <ccts:CardinalityText></ccts:CardinalityText>
  <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
  <ccts:PropertyTermName></ccts:PropertyTermName>
  <ccts:QualifierTermName></ccts:QualifierTermName>
  <ccts:UsageRuleText></ccts:UsageRuleText>
  <ccts:BusinessTermName></ccts:BusinessTermName>
  <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
  <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
  <ccts:ProductContext></ccts:ProductContext>
  <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
  <ccts:CategoryCode>ASBIE</ccts:CategoryCode>
  <ccts:DictionaryEntryName></ccts:DictionaryEntryName>
  <ccts:VersionID></ccts:VersionID>
  <ccts:DefinitionText></ccts:DefinitionText>
  <ccts:CardinalityText></ccts:CardinalityText>
  <ccts:ObjectClassTermName></ccts:ObjectClassTermName>
  <ccts:PropertyTermName></ccts:PropertyTermName>
  <ccts:QualifierTermName></ccts:QualifierTermName>
  <ccts:UsageRuleText></ccts:UsageRuleText>
  <ccts:BusinessTermName></ccts:BusinessTermName>
  <ccts:GeopoliticalOrRegionContext></ccts:GeopoliticalOrRegionContext>
  <ccts:OfficialConstraintContext></ccts:OfficialConstraintContext>
  <ccts:ProductContext></ccts:ProductContext>
</xs:documentation>
Appendix C. Naming & Design Rules List

[R 1] CONFORMANCE SHALL BE DETERMINED THROUGH ADHERENCE TO THE CONTENT OF NORMATIVE SECTIONS, RULES AND DEFINITIONS.

[R 2] UN/CEFACT SCHEMA MUST FOLLOW THE STANDARD STRUCTURE DEFINED IN APPENDIX A.

[R 3] EACH ELEMENT OR ATTRIBUTE XML NAME MUST HAVE ONE AND ONLY ONE FULLY QUALIFIED XPATH (FQXP).

[R 4] ELEMENT, ATTRIBUTE AND TYPE NAMES MUST BE IN THE ENGLISH LANGUAGE, USING THE PRIMARY ENGLISH SPELLINGS PROVIDED IN THE OXFORD ENGLISH DICTIONARY.

[R 5] LOWER-CAMEL-CASE (LCC) MUST BE USED FOR NAMING ATTRIBUTES.

[R 6] UPPER-CAMEL-CASE (UCC) MUST BE USED FOR NAMING ELEMENTS AND TYPES.

[R 7] NAMES MUST BE IN SINGULAR FORM UNLESS THE CONCEPT ITSELF IS PLURAL.

[R 8] NAMES MUST NOT CONTAIN NON-LETTER CHARACTERS, UNLESS REQUIRED BY LANGUAGE-SPECIFIC RULES.

[R 9] XML NAMES CONSTRUCTED FROM DICTIONARY ENTRY NAMES MUST NOT INCLUDE PERIODS, SPACES, OR OTHER SEPARATORS; OR CHARACTERS NOT ALLOWED BY W3C XML 1.0 FOR XML NAMES.

[R 10] ELEMENT AND XSD:SIMPLE AND XSD:COMPLEXTYPE NAMES MUST NOT USE ACRONYMS, ABBREVIATIONS, OR OTHER WORD TRUNCATIONS, EXCEPT THOSE INCLUDED IN THE UN/CEFACT CONTROLLED VOCABULARY.

[R 11] ALL ELEMENT DECLARATIONS FOR BBIES AND ASBIES MUST BE LOCALLY DECLARED WITHIN THE PARENT ABIE TYPE.


[R 13] A ROOT SCHEMA MUST BE CREATED FOR EACH UNIQUE BUSINESS INFORMATION EXCHANGE.

[R 14] A ROOT SCHEMA MUST NOT REPLICATE REUSABLE CONSTRUCTS AVAILABLE IN SCHEMA MODULES CAPABLE OF BEING REFERENCED THROUGH XSD:INCLUDE OR XSD:IMPORT.
UN/CEFACT SCHEMA MODULES MUST EITHER BE TREATED AS EXTERNAL SCHEMA
MODULES OR AS INTERNAL SCHEMA MODULES OF THE ROOT SCHEMA.

ROOT SCHEMAS MUST ONLY DEFINE A SINGLE COMPLEX TYPE THAT FULLY
DESCRIBES THE BUSINESS INFORMATION EXCHANGE.

ALL UN/CEFACT INTERNAL SCHEMA MODULES MUST BE IN THE SAME NAMESPACE
AS THEIR CORRESPONDING RSM:ROOTSCHEMA.

EACH UN/CEFACT INTERNAL SCHEMA MODULE MUST BE NAMED
(PARENTROOTSCHEMAFILENAME) (INTERNALSCHMAFUNCTION) SCHEMA MODULE

A CORE COMPONENT TYPE SCHEMA MODULE MUST BE CREATED
THE CCT:CORECOMPONENTTYPE SCHEMA MODULE MUST BE NAMED "CCTS CCT
SCHEMA MODULE"

AN UNQUALIFIED DATA TYPE SCHEMA MODULE MUST BE CREATED
THE UDT:UNQUALIFIEDDATATYPE SCHEMA MODULE MUST BE NAMED "UN/CEFACT UNQUALIFIED DATA TYPE SCHEMA MODULE"

A QUALIFIED DATA TYPE SCHEMA MODULE MUST BE CREATED
THE QDT:QUALIFIEDDATATYPE SCHEMA MODULE MUST BE NAMED "UN/CEFACT QUALIFIED DATA TYPE SCHEMA MODULE"

AN AGGREGATE BUSINESS INFORMATION ENTITY SCHEMA MODULE MUST BE CREATED

THE RAM:REUSABLEAGGREGATEBUSINESSINFORMATIONENTITY SCHEMA MODULE MUST BE NAMED "UN/CEFACT AGGREGATE BUSINESS INFORMATION ENTITY SCHEMA MODULE"

REUSABLE CODE LIST SCHEMA MODULES MUST BE CREATED TO CONVEY CODE LIST ENUMERATIONS


AN IDENTIFIER LIST SCHEMA MODULE MUST BE CREATED FOR EACH IDENTIFIER LIST THAT MIMICS CODE LIST FUNCTIONALITY TO CONVEY ENUMERATIONS OF THE IDENTIFIER LIST VALUE AND TOKEN FOR THAT VALUE

THE NAME OF EACH UN/CEFACT IDENTIFIER LIST SCHEMA MODULE MUST BE OF THE FORM: <IDENTIFICATION SCHEME AGENCY NAME><IDENTIFICATION SCHEME NAME> - IDENTIFIER LIST SCHEMA MODULE WHERE: IDENTIFICATION SCHEME AGENCY NAME = AGENCY THAT MAINTAINS THE IDENTIFIER LIST IDENTIFICATION SCHEME NAME = NAME AS ASSIGNED BY THE AGENCY THAT MAINTAINS THE IDENTIFIER LIST
IMPORTED SCHEMA MODULES MUST BE FULLY CONFORMANT WITH UN/CEFACT NAMING AND DESIGN RULES AND THE CORE COMPONENTS TECHNICAL SPECIFICATION.

EVERY UN/CEFACT DEFINED OR USED SCHEMA MODULE MUST HAVE A NAMESPACE DECLARED, USING THE XSD:TARGETNAMESPACE ATTRIBUTE.

EVERY DEFINED OR USED SCHEMA MODULE VERSION OTHER THAN INTERNAL SCHEMA MODULES MUST HAVE ITS OWN UNIQUE NAMESPACE.

UN/CEFACT PUBLISHED NAMESPACE DECLARATIONS OR CONTENTS MUST NEVER BE CHANGED.

UN/CEFACT NAMESPACES MUST BE DEFINED AS URNS.

THE NAMES FOR NAMESPACES MUST HAVE THE FOLLOWING STRUCTURE WHILE THE SCHEMAS ARE AT DRAFT STATUS:

WHERE: SCHEMATYPE = A TOKEN IDENTIFYING THE TYPE OF SCHEMA MODULE:
DATA|PROCESS|CODELIST
NAME = THE NAME OF THE MODULE (USING UNDERSCORE AS SEPARATOR) MAJOR = THE MAJOR VERSION NUMBER, SEQUENTIALLY ASSIGNED, FIRST RELEASE STARTING WITH THE NUMBER 1. MINOR = THE MINOR VERSION NUMBER WITHIN A MAJOR RELEASE, SEQUENTIALLY ASSIGNED, FIRST RELEASE STARTING WITH THE NUMBER 0. NOT APPLICABLE FOR CODELIST SCHEMA.
REVISION = SEQUENTIALLY ASSIGNED ALPHANUMERIC CHARACTER FOR EACH REVISION OF A MINOR RELEASE. ONLY APPLICABLE WHERE STATUS = DRAFT AND SCHEMA TYPE DOES NOT EQUAL CODELIST.

THE NAMESPACE NAMES FOR SCHEMAS HOLDING SPECIFICATION STATUS MUST BE OF THE FORM:

WHERE: SCHEMATYPE = A TOKEN IDENTIFYING THE TYPE OF SCHEMA MODULE:
DATA|PROCESS|CODELIST
NAME = THE NAME OF THE MODULE (USING UNDERSCORE AS SEPARATOR) MAJOR = THE MAJOR VERSION NUMBER, SEQUENTIALLY ASSIGNED, FIRST RELEASE STARTING WITH THE NUMBER 1. MINOR = THE MINOR VERSION NUMBER WITHIN A MAJOR RELEASE, SEQUENTIALLY ASSIGNED, FIRST RELEASE STARTING WITH THE NUMBER 0. NOT APPLICABLE FOR CODELIST SCHEMA.

UN/CEFACT NAMESPACES MUST ONLY CONTAIN UN/CEFACT DEVELOPED SCHEMA MODULES.

THE GENERAL STRUCTURE FOR SCHEMA LOCATION MUST BE:

HTTP://WWW.UNECE.ORG/UNCEFACT/<SCHEMATYPE>/<NAME>_<MAJOR>_<MINOR>_<REVISION>_<STATUS>.XSD WHERE: SCHEMATYPE = A TOKEN IDENTIFYING THE TYPE OF SCHEMA MODULE: DATA|PROCESS|CODELIST NAME = THE NAME OF THE MODULE (USING UNDERSCORE AS SEPARATOR) MAJOR = THE MAJOR VERSION NUMBER, SEQUENTIALLY ASSIGNED, FIRST RELEASE STARTING WITH THE NUMBER 1. MINOR = THE MINOR VERSION NUMBER WITHIN A MAJOR RELEASE, SEQUENTIALLY ASSIGNED, FIRST RELEASE STARTING WITH THE NUMBER 0. REVISION = SEQUENTIALLY ASSIGNED ALPHANUMERIC CHARACTER FOR EACH REVISION OF A MINOR RELEASE. ONLY APPLICABLE WHERE STATUS = DRAFT.
STATUS = THE STATUS OF THE SCHEMA AS: DRAFT|STANDARD

EACH XSD:SCHEMALLOCATION ATTRIBUTE DECLARATION MUST CONTAIN A PERSISTENT AND RESOLVABLE URL.
Each XSD:schemaLocation attribute declaration URL must contain an absolute path.

Schema modules must be located under the directory:

http://www.unece.org/uncefact/cc/schema/<schema-mod-name>.xsd

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

Every schema major version must have the URI of:
urn:un:unece:uncefact:<schematype>:<status>:<name>:<major>:0:[<revision>]

Every UN/CEFACT schema and schema module major version number must be a sequentially assigned incremental integer greater than zero.

Minor versioning must be limited to declaring new optional XSD constructs, extending existing XSD constructs and refinements of an optional nature.

Every UN/CEFACT schema minor version must have the URI of:

For UN/CEFACT minor version changes, the name of the schema construct must not change.

Changes in minor versions must not break semantic compatibility with prior versions.

UN/CEFACT minor version schema must incorporate all XML constructs from the immediately preceding major or minor version schema.

All UN/CEFACT schema design rules must be based on the W3C XML schema recommendations: XML schema part 1: structures and XML schema part 2: datatypes.

All schemata and instances must be based on the W3C suite of technical specifications holding recommendation status.

Element form default must be declared as qualified.

Attribute form default must be declared as unqualified.

The "xsd" prefix must be used.


The xsi prefix shall be used where appropriate.
[R 57] PROCESSING INSTRUCTIONS MUST NOT BE USED.

[R 58] NOTATIONS MUST NOT BE USED.

[R 59] WILDCARDS MUST NOT BE USED.

[R 60] THE xsd: ANY ELEMENT MUST NOT BE USED.

[R 61] THE xsd: ANY ATTRIBUTE MUST NOT BE USED.

[R 62] MIXED CONTENT MUST NOT BE USED (EXCLUDING DOCUMENTATION).

[R 63] SUBSTITUTION GROUPS MUST NOT BE USED.

[R 64] ID/IDREF MUST NOT BE USED.

[R 65] KEY/KEYREF MUST BE USED FOR INFORMATION ASSOCIATION.

[R 66] THE ABSENCE OF A CONSTRUCT OR DATA MUST NOT CARRY MEANING.

[R 67] USER DEFINED ATTRIBUTES MUST ONLY BE USED TO CONVEY CORE COMPONENT TYPE (CCT) SUPPLEMENTARY COMPONENT INFORMATION.

[R 68] THE nILLABLE ATTRIBUTE MUST NOT BE USED.

[R 69] ALL ELEMENT DECLARATIONS MUST BE LOCAL EXCEPT FOR A ROOT ELEMENT THAT MUST BE DECLARED GLOBALLY.

[R 70] EMPTY ELEMENTS MUST NOT BE USED.

[R 72] THE ELEMENT DECLARATION OF xsi: NIL MUST NOT APPEAR IN ANY CONFORMING INSTANCE.

[R 73] THE xsd: ALL ELEMENT MUST NOT BE USED.

[R 74] MIXED-CONTENT ELEMENTS MUST NOT BE USED.

[R 75] ALL TYPE DEFINITIONS MUST BE NAMED.

[R 76] TYPE DEFINITIONS MUST NOT DUPLICATE THE FUNCTIONALITY OF EXISTING xsd: BUILT-IN SIMPLE TYPES.

[R 77] USER DEFINED SIMPLE TYPES DEFINITIONS MUST NOT BE USED.

[R 78] EXTENSION MUST ONLY BE USED IN THE cct: CORECOMPONENTTYPES SCHEMA MODULE AND THE udt: UNQUALIFIEDDATATYPE SCHEMA MODULE. WHEN USED IT MUST ONLY EXTEND A BUILT-IN xsd: DATATYPE.
THE ROOT SCHEMA MODULE MUST BE REPRESENTED BY THE TOKEN “RSM”.

THE RSM:ROOTSCHEMA MUST IMPORT THE FOLLOWING SCHEMA MODULES:
- RAM:REUSABLEABIE SCHEMA MODULE
- QDT:QUALIFIEDDATATYPES MODULE

A RSM:ROOTSCHEMA IN ONE UN/CEFACT NAMESPACE THAT IS DEPENDENT UPON TYPE DEFINITIONS OR ELEMENT DECLARATION DEFINED IN ANOTHER NAMESPACE MUST IMPORT THE RSM:ROOTSCHEMA FROM THAT NAMESPACE.

A RSM:ROOTSCHEMA IN ONE UN/CEFACT NAMESPACE THAT IS DEPENDANT UPON TYPE DEFINITIONS OR ELEMENT DECLARATIONS DEFINED IN ANOTHER NAMESPACE MUST NOT IMPORT SCHEMA MODULES FROM THAT NAMESPACE OTHER THEN THE RSM:ROOTSCHEMA.

THE RSM:ROOTSCHEMA MUST INCLUDE ANY INTERNAL SCHEMA MODULES THAT RESIDE IN THE ROOT SCHEMA NAMESPACE.

A SINGLE GLOBAL ELEMENT KNOWN AS THE ROOT ELEMENT MUST BE GLOBALLY DECLARED IN A RSM:ROOTSCHEMA.

THE ROOT ELEMENT MUST REFLECT THE MESSAGE ASSEMBLY THAT IS DEFINED FOR THE ACTUAL CONTENT OF THE BUSINESS INFORMATION.

THE NAME OF THE ROOT ELEMENT MUST BE THE NAME OF THE MESSAGE ASSEMBLY WITH SEPARATORS AND SPACES REMOVED.

FOR THE ROOT ELEMENT A CORRESPONDING COMPLEX TYPE THAT REPRESENTS THE MESSAGE ASSEMBLY MUST BE DEFINED.

THE NAME OF THE TOP-LEVEL COMPLEX TYPE MUST BE THE NAME OF THE ROOT ELEMENT WITH THE WORD “TYPE” APPENDED.

FOR EVERY RSM:ROOTSCHEMA ROOT ELEMENT DECLARATION A STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:

ALL UN/CEFACT INTERNAL SCHEMA MODULES MUST BE IN THE SAME NAMESPACE AS THEIR CORRESPONDING RSM:ROOTSCHEMA.

THE INTERNAL SCHEMA MODULE MUST BE REPRESENTED BY THE SAME TOKEN AS ITS RSM:ROOTSCHEMA.

THE SCHEMA MODULE MUST BE REPRESENTED BY THE TOKEN “RAM”.

THE RAM:REUSABLEAGGREGATEBUSINESSINFORMATIONENTITY SCHEMA MUST IMPORT THE FOLLOWING SCHEMA MODULES:
- UDT:UNQUALIFIEDDATATYPE MODULE

FOR EVERY OBJECT CLASS (ABIE) IDENTIFIED IN THE UN/CEFACT SYNTAX-NEUTRAL MODEL, A NAMED XSD:COMPLEXTYPE MUST BE DEFINED.

[R 96] EVERY AGGREGATE BUSINESS INFORMATION ENTITY (ABIE) XSD:COMPLEXTYPE DEFINITION XSD:CONTENT MODEL MUST USE THE XSD:SEQUENCE AND/OR XSD:CHOICE ELEMENTS WITH APPROPRIATE LOCAL ELEMENT DECLARATIONS TO REFLECT EACH PROPERTY (BBIE OR ASBIE) OF ITS CLASS.

[R 97] RECURSION OF XSD:SEQUENCE AND/OR XSD:CHOICE MUST NOT OCCUR.


[R 99] FOR EVERY ATTRIBUTE OF AN OBJECT CLASS (BBIE) IDENTIFIED IN THE UN/CEFACT SYNTAX-NEUTRAL MODEL, A NAMED XSD:ELEMENT MUST BE LOCALLY DECLARED WITHIN THE XSD:COMPLEXTYPE REPRESENTING THE ABIE.


[R 101] THE BBIE ELEMENT MUST BE BASED ON AN APPROPRIATE DATA TYPE THAT IS DEFINED IN THE UN/CEFACT QDT:QUALIFIEDDATATYPE OR UDT:UNQUALIFIEDDATATYPE SCHEMA MODULES.

[R 102] FOR EVERY ASSOCIATION (ASBIE) IDENTIFIED IN THE UN/CEFACT SYNTAX-NEUTRAL MODEL, A NAMED XSD:ELEMENT MUST BE LOCALLY DECLARED WITHIN THE XSD:COMPLEXTYPE REPRESENTING THE ABIE.

[R 103] EACH ASBIE ELEMENT NAME DECLARATION MUST BE BASED ON THE PROPERTY TERM AND OBJECT CLASS OF THE ASSOCIATION BUSINESS INFORMATION ENTITY (ASBIE). IF THERE ARE DUPLICATE WORDS IN THE PROPERTY TERM AND REPRESENTATION TERMS OF THE SOURCE DICTIONARY ENTRY NAME, THEN THE DUPLICATE WORDS MUST BE REMOVED.

[R 104] THE ELEMENT REPRESENTING AN ASSOCIATION BUSINESS INFORMATION ENTITY (ASBIE) MUST BE OF THE COMPLEX TYPE CORRESPONDING TO ITS ASSOCIATED AGGREGATE BUSINESS INFORMATION (ABIE).

[R 105] EACH XSD:COMPLEXTYPE AND XSD:ELEMENT DEFINITION MUST USE THE XSD:ANNOTATION ELEMENT.


[R 107] FOR EVERY ABIE XSD:COMPLEXTYPE DEFINITION A STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:

[R 108] FOR EVERY BBIE XSD:ELEMENT DECLARATION A STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:
FOR EVERY ASBIE XSD:ELEMENT DECLARATION A STRUCTURED SET OF
ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:

THE CORE COMPONENT TYPE (CCT) SCHEMA MODULE MUST BE REPRESENTED BY
THE TOKEN "CCT".

EVERY CCT:CORECOMPONENTTYPE MUST BE DEFINED AS A NAMED XSD:COMPLEXTYPE
IN THE CCT:CORECOMPONENTTYPE SCHEMA MODULE.

THE NAME OF EACH XSD:COMPLEXTYPE BASED ON A CCT:CORECOMPONENTTYPE
MUST BE THE DICTIONARY ENTRY NAME OF THE CORE COMPONENT TYPE (CCT), WITH THE
SEPARATORS AND SPACES REMOVED.

EACH CCT:CORECOMPONENTTYPE XSD:COMPLEXTYPE DEFINITION MUST CONTAIN ONE
XSD:SIMPLECONTENT ELEMENT.

THE CCT:CORECOMPONENTTYPE XSD:COMPLEXTYPE DEFINITION MUST CONTAIN ONE
XSD:EXTENSION ELEMENT. THIS XSD:EXTENSION ELEMENT
MUST INCLUDE AN XSD BASED ATTRIBUTE THAT DEFINES THE SPECIFIC XSD:BUILT-IN DATA
TYPE REQUIRED FOR THE CCT CONTENT COMPONENT.

WITHIN THE CCT:CORECOMPONENTTYPE XSD:EXTENSION ELEMENT A XSD:ATTRIBUTE
MUST BE DECLARED FOR EACH SUPPLEMENTARY COMPONENT PERTAINING TO THAT
CCT:CORECOMPONENTTYPE.

A XSD:ATTRIBUTE MUST BE DECLARED FOR EACH SUPPLEMENTARY COMPONENT.

EACH SUPPLEMENTARY COMPONENT XSD:ATTRIBUTE "NAME" MUST BE THE
SUPPLEMENTARY COMPONENT NAME WITH THE SEPARATORS AND SPACES REMOVED.

EACH XSD:COMPLEXTYPE DEFINITION MUST USE THE XSD:ANNOTATION ELEMENT.

AN XSD:ANNOTATION ELEMENT DECLARATION MUST APPEAR IMMEDIATELY AFTER
THE STARTING TAG OF THE XSD:COMPLEXTYPE.

FOR EVERY CCT:CORECOMPONENTTYPE XSD:COMPLEXTYPE DEFINITION A
STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:

FOR EVERY SUPPLEMENTARY COMPONENT XSD:ATTRIBUTE DECLARATION A
STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:

THE UNQUALIFIED DATA TYPES SCHEMA MODULE NAMESPACE MUST BE
REPRESENTED BY THE TOKEN "UDT".

THE UDT:UNQUALIFIEDDATATYPE SCHEMA MUST IMPORT THE FOLLOWING SCHEMA
MODULES: – IDS:IDENTIFIERLIST SCHEMA MODULES – CLM:CODELIST SCHEMA MODULES

A UDT:UNQUALIFIEDDATATYPE MUST BE DEFINED FOR EACH APPROVED PRIMARY
AND SECONDARY REPRESENTATION TERMS IDENTIFIED IN THE CCTS PERMISSIBLE
REPRESENTATION TERMS TABLE.


EVERY UDT:UNQUALIFIEDDATATYPE XSD:COMPLEXTYPE DEFINITION MUST CONTAIN ONE XSD:SIMPLECTENT ELEMENT.


WITHIN THE UDT:UNQUALIFIEDDATATYPE XSD:COMPLEXTYPE XSD:EXTENSION ELEMENT AN XSD:ATTRIBUTE MUST BE DECLARED FOR EACH SUPPLEMENTARY COMPONENT PERTAINING TO THE UNDERLYING CCT, UNLESS THE ATTRIBUTE IS CONTAINED IN THE NAMESPACE DECLARATION.


EACH SUPPLEMENTARY COMPONENT XSD:ATTRIBUTE NAME MUST BE THE SUPPLEMENTARY COMPONENT NAME WITH THE SEPARATORS AND SPACES REMOVED.

EACH UDT:UNQUALIFIEDDATATYPE XSD:COMPLEXTYPE AND XSD:SIMPLECTYPE DEFINITION MUST USE THE XSD:ANNOTATION ELEMENT.

AN XSD:ANNOTATION ELEMENT DECLARATION MUST APPEAR IMMEDIATELY AFTER THE STARTING TAG OF THE XSD:COMPLEXTYPE OR XSD:SIMPLECTYPE.

FOR EVERY UDT:UNQUALIFIEDDATATYPE XSD:COMPLEXTYPE OR XSD:SIMPLECTYPE DEFINITION A STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:
FOR EVERY SUPPLEMENTARY COMPONENT XSD:ATTRIBUTE DECLARATION A
STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:

THE UN/CEFACT:QUALIFIEDDATATYPE SCHEMA MODULE NAMESPACE MUST BE
REPRESENTED BY THE Token “QDT”.

THE QDT:QUALIFIEDDATATYPE SCHEMA MODULE MUST IMPORT THE –
UDT:UNQUALIFIEDDATATYPE SCHEMA MODULE – IDS:IDENTIFIERLIST SCHEMA MODULES –
CLM:CODELIST SCHEMA MODULE

WHERE REQUIRED TO CHANGE FACETS OF AN EXISTING
UDT:UNQUALIFIEDDATATYPE, A NEW DATA TYPE MUST BE DEFINED IN THE
QDT:QUALIFIEDDATATYPE SCHEMA MODULE.

IF A QDT:QUALIFIEDDATATYPE IS BASED ON AN UDT:UNQUALIFIEDDATATYPE THAT IS
AN XSD:SIMPLETEYPE, THE QDT:QUALIFIEDDATATYPE MUST ALSO BE AN XSD:SIMPLETEYPE.

IF A QDT:QUALIFIEDDATATYPE IS BASED ON AN UDT:UNQUALIFIEDDATATYPE THAT IS
A XSD:COMPLEXTYPE, THE QDT:QUALIFIEDDATATYPE MUST ALSO BE AN XSD:COMPLEXTYPE.

IF A QDT:QUALIFIEDDATATYPE’S RESTRICTIONS MAP DIRECTLY TO THE
PROPERTIES OF A XSD:BUILT-IN DATA TYPE, THE QDT:QUALIFIEDDATATYPE MUST BE
DEFINED AS A NAMED XSD:SIMPLETEYPE IN THE QDT:QUALIFIEDDATATYPE SCHEMA MODULE.

THE NAME OF A QDT:QUALIFIEDDATATYPE MUST BE THE DICTIONARY ENTRY NAME
OF ITS BASE UDT:UNQUALIFIEDDATATYPE WITH AN APPROPRIATE QUALIFIER OR QUALIFIERS.

EACH QDT:QUALIFIEDDATATYPE XSD:COMPLEXTYPE DEFINITION MUST CONTAIN ONE
XSD:SIMPLETECONTENT ELEMENT.

THE QDT:QUALIFIEDDATATYPE XSD:COMPLEXTYPE DEFINITION XSD:SIMPLETECONTENT
ELEMENT MUST CONTAIN ONE XSD:RESTRICTION ELEMENT WITH AN XSD:BASE ATTRIBUTE
WHOSE VALUE IS EQUAL TO AN UDT:UNQUALIFIEDDATATYPE OR AN EXTERNALLY IMPORTED
CODE LIST.

THE QDT:QUALIFIEDDATATYPE XSD:COMPLEXTYPE DEFINITION XSD:SIMPLETECONTENT
ELEMENT MUST ONLY RESTRICT ATTRIBUTES DECLARED IN ITS BASE TYPE, OR MUST ONLY
RESTRICT FACETS EQUIVALENT TO ALLOWED SUPPLEMENTARY COMPONENTS.

EACH QDT:QUALIFIEDDATATYPE XSD:COMPLEXTYPE AND XSD:SIMPLETEYPE
DEFINITION MUST USE THE XSD:ANNOTATION ELEMENT.

AN XSD:ANNOTATION ELEMENT DECLARATION MUST APPEAR IMMEDIATELY AFTER
THE STARTING TAG OF THE XSD:COMPLEXTYPE OR XSD:SIMPLETEYPE.

EVERY QDT:QUALIFIEDDATATYPE DEFINITION MUST CONTAIN A STRUCTURED SET
OF ANNOTATIONS IN THE FOLLOWING SEQUENCE AND PATTERN:

FOR EVERY SUPPLEMENTARY COMPONENT XSD:ATTRIBUTE DECLARATION A
STRUCTURED SET OF ANNOTATIONS MUST BE PRESENT IN THE FOLLOWING PATTERN:
INTERNAL CODE LISTS MUST NOT DUPLICATE EXISTING EXTERNAL CODE LISTS WHERE THEY ARE AVAILABLE IN A SCHEMA MODULE FORM THAT IS CAPABLE OF BEING IMPORTED.

EACH UN/CEFACT MAINTAINED CODE LIST MUST BE DEFINED IN ITS OWN SCHEMA MODULE.

EACH UN/CEFACT MAINTAINED CODE LIST SCHEMA MODULE MUST BE REPRESENTED BY A UNIQUE TOKEN CONSTRUCTED AS FOLLOWS: CLM[QUALIFIED DATA TYPE NAME]<CODE LIST. AGENCY. IDENTIFIER|CODE LIST. AGENCY NAME. TEXT><CODE LIST. IDENTIFICATION. IDENTIFIER|CODE LIST. NAME. TEXT> WITH ANY REPEATED WORDS ELIMINATED.

CODE LIST SCHEMA MODULES MUST NOT IMPORT OR INCLUDE ANY OTHER SCHEMA MODULES.

FOR THE TOP-LEVEL ELEMENT A CORRESPONDING TOP-LEVEL XSD:COMPLEXTYPE MUST BE DEFINED.

THE NAME FOR THE XSD:COMPLEXTYPE MUST BE BASED ON THE VALUE OF THE CODE LIST. NAME. TEXT WITH THE WORD “TYPE” APPENDED.

EACH XSD:COMPLEXTYPE MUST CONTAIN ONE XSD:SIMPLECTYPE ELEMENT.

THE XSD:COMPLEXTYPE DEFINITION XSD:SIMPLECTYPE ELEMENT MUST CONTAIN ONE XSD:EXTENSION ELEMENT. THIS XSD:EXTENSION ELEMENT MUST INCLUDE AN XSD BASED ATTRIBUTE THAT DEFINES THE SPECIFIC LIST CONTAINING THE ACTUAL CODE VALUES.

WITHIN EACH CODE LIST MODULE ONE, AND ONLY ONE, NAMED XSD:SIMPLECTYPE MUST BE DEFINED FOR THE CONTENT COMPONENT.


THE XSD:RESTRICTION ELEMENT BASE ATTRIBUTE VALUE MUST BE SET TO “XSD:TOKEN”.

EACH CODE IN THE CODE LIST MUST BE EXPRESSED AS AN XSD:ENUMERATION, WHERE THE XSD:VALUE FOR THE ENUMERATION IS THE ACTUAL CODE VALUE.

THE XSD FACET FEATURE MUST NOT BE USED IN THE CODE LIST SCHEMA MODULE.

FOR EACH CODE LIST A SINGLE ROOT ELEMENT MUST BE GLOBALLY DECLARED.

THE NAME OF ROOT ELEMENT MUST BE BASED ON THE CODE LIST. NAME. TEXT FOLLOWING THE NAMING RULES AS DEFINED IN SECTION 2.3.
[R 167] THE ROOT ELEMENT MUST BE OF A TYPE REPRESENTING THE ACTUAL LIST OF CODE VALUES.

[R 168] EACH XSD:ENUMERATION MUST INCLUDE AN ANNOTATION DOCUMENTATION PROVIDING THE CODE NAME AND THE CODE DESCRIPTION.

[R 169] WHERE AN IDENTIFICATION SCHEME IS IDENTIFIED (CONTAINS AN IDENTIFIER, A TOKEN FOR THE IDENTIFIER, AND AN Optionally, A DESCRIPTION) TO HAVE THE SAME FUNCTIONALITY AS A CODE LIST AND A BUSINESS REQUIREMENT EXISTS FOR ENUMERATING THE IDENTIFIER LIST, A SCHEMA MODULE OF THE SAME FORM AS THE CODE LIST SCHEMA MUST BE CREATED.

[R 170] INTERNAL IDENTIFIER LISTS MUST NOT DUPLICATE EXISTING EXTERNAL IDENTIFIER LISTS WHERE THEY ARE AVAILABLE IN A SCHEMA MODULE FORM THAT IS CAPABLE OF BEING IMPORTED.

[R 171] EACH UN/CEFACT MAINTAINED IDENTIFIER LIST MUST BE DEFINED IN ITS OWN SCHEMA MODULE.

[R 172] EACH UN/CEFACT MAINTAINED IDENTIFIER LIST SCHEMA MODULE MUST BE REPRESENTED BY A UNIQUE TOKEN CONSTRUCTED AS FOLLOWS: IDS[QUALIFIED DATA TYPE NAME]<IDENTIFICATION SCHEME AGENCY. IDENTIFIER><IDENTIFICATION SCHEME. IDENTIFIER>

[R 173] IDENTIFIER LIST SCHEMA MODULES MUST NOT IMPORT OR INCLUDE ANY OTHER SCHEMA MODULES.

[R 174] WITHIN EACH IDENTIFIER LIST MODULE ONE, AND ONLY ONE, NAMED XSD:SIMPLETEYPE MUST BE DEFINED FOR THE CONTENT COMPONENT.

[R 175] THE NAME OF THE XSD:SIMPLETEYPE MUST BE THE NAME OF ROOT ELEMENT WITH THE WORD “CONTENTTYPE” APPENDED.

[R 176] THE XSD:RESTRICTION ELEMENT BASE ATTRIBUTE VALUE MUST BE SET TO “XSD:TOKEN”.

[R 177] EACH IDENTIFIER IN THE IDENTIFIER LIST MUST BE EXPRESSED AS AN XSD:ENUMERATION, WHERE THE XSD:VALUE FOR THE ENUMERATION IS THE ACTUAL IDENTIFIER VALUE.

[R 178] THE XSD FACET FEATURE MUST NOT BE USED IN THE IDENTIFIER LIST SCHEMA MODULE.

[R 179] FOR EACH IDENTIFIER LIST A SINGLE ROOT ELEMENT MUST BE GLOBALLY DECLARED.

[R 180] THE NAME OF THE ROOT ELEMENT MUST BE BASED ON THE IDENTIFICATION SCHEME NAME. TEXT FOLLOWING THE NAMING RULES AS DEFINED IN SECTION 2.3.

[R 181] THE ROOT ELEMENT MUST BE OF A TYPE REPRESENTING THE ACTUAL LIST OF IDENTIFIER VALUES.
Each XSD:Enumeration must include an annotation documentation providing the identifier name and optionally the description of the identifier.

All UN/CEFACT XML must be instantiated using UTF. UTF-8 should be used as the preferred encoding. If UTF-8 is not used, UTF-16 must be used.

The xsi:type attribute must not be used.
Appendix D. Definition of Terms

**Aggregate Business Information Entity (ABIE)** – A collection of related pieces of business information that together convey a distinct business meaning in a specific Business Context. Expressed in modelling terms, it is the representation of an Object Class, in a specific Business Context.

**Aggregate Core Component - (ACC)** – A collection of related pieces of business information that together convey a distinct business meaning, independent of any specific Business Context. Expressed in modelling terms, it is the representation of an Object Class, independent of any specific Business Context.

**Assembly Rules** - Assembly Rules group sets of unrefined Business Information Entities into larger structures. Assembly Rules are more fully defined and explained in the Assembly Rules Supplemental Document.

**Association Business Information Entity (ASBIE)** - A Business Information Entity that represents a complex business characteristic of a specific Object Class in a specific Business Context. It has a unique Business Semantic definition. An Association Business Information Entity represents an Association Entity, which describes its structure. An Association Business Information Entity is derived from an Association Core Component.

**Association Business Information Entity Property** - A Business Information Entity Property for which the permissible values are expressed as a complex structure, represented by an Aggregate Business Information Entity.

**Association Core Component (ASCC)** - A Core Component which constitutes a complex business characteristic of a specific Aggregate Core Component that represents an Object Class. It has a unique Business Semantic definition. An Association Core Component represents an Association Core Component Property and is associated to an Aggregate Business Information Entity, which describes its structure.

**Association Core Component Property** – A Component Property for which the permissible values are expressed as a complex structure, represented by an Aggregate Core Component.

**Attribute** – A named value or relationship that exists for some or all instances of some entity and is directly associated with that instance.

**Basic Business Information Entity (BBIE)** – A Business Information Entity that represents a singular business characteristic of a specific Object Class in a specific Business Context. It has a unique Business Semantic definition. A Basic Business Information Entity represents a Basic Business Information Entity Property and is therefore linked to a Data Type, which describes it values. A Basic Business Information Entity is derived from a Basic Core Component.

**Basic Business Information Entity Property** – A Business Information Entity Property for which the permissible values are expressed by simple values, represented by a Data Type.

**Basic Core Component (BCC)** – A Core Component which constitutes a singular business characteristic of a specific Aggregate Core Component that represents a Object Class. It has a unique Business Semantic definition. A Basic Core Component represents a Basic Core Component Property and is therefore of a Data Type, which defines its set of values. Basic Core Components function as the properties of Aggregate Core Components.

**Basic Core Component (CC) Property** – A Component Property for which the permissible values are expressed by simple values, represented by a Data Type.

**Business Context** – The formal description of a specific business circumstance as identified by the values of a set of Context Categories, allowing different business circumstances to be uniquely distinguished.

**Business Information Entity (BIE)** – A piece of business data or a group of pieces of business data with a unique Business Semantic definition. A Business Information Entity can be a Basic Business Information Entity (BBIE), an Association Business Information Entity (ASBIE), or an Aggregate Business Information Entity (ABIE).

**Business Information Entity (BIE) Property** – A business characteristic belonging to the Object Class in its specific Business Context that is represented by an Aggregate Business Information Entity.

**Business Libraries** – A collection of approved process models specific to a line of business (e.g., shipping, insurance).


**Business Process Context** – The Business Process name(s) as described using the UN/CEFACT Catalogue of Common Business Processes as extended by the user.
Business Process Role Context – The actors conducting a particular Business Process, as identified in
the UN/CEFACT Catalogue of Common Business Processes.
Business Semantic(s) – A precise meaning of words from a business perspective.
Business Term – This is a synonym under which the Core Component or Business Information Entity is
commonly known and used in the business. A Core Component or Business Information Entity may have
several Business Terms or synonyms.
Cardinality – An indication whether a characteristic is optional, mandatory and/or repetitive.
Catalogue of Business Information Entities – This represents the approved set of Business
Information Entities from which to choose when applying the Core Component discovery process.
Catalogue of Core Components – see Core Component Catalogue.
CCL – see Core Component Library.
Child Core Component – A Core Component used as part of a larger aggregate construct.
Classification Scheme – This is an officially supported scheme to describe a given Context Category.
Constraint Language – A formal expression of actions occurring in specific Contexts to assemble,
structurally refine, and semantically qualify Core Components. The result of applying the Constraint
Language to a set of Core Components in a specific Context is a set of Business Information Entities.
Content Component – Defines the Primitive Type used to express the content of a Core Component
Type.
Content Component Restrictions – The formal definition of a format restriction that applies to the
possible values of a Content Component.
Context – Defines the circumstances in which a Business Process may be used. This is specified by a
set of Context Categories known as Business Context.
Context Category – A group of one or more related values used to express a characteristic of a business
circumstance.
Context Rules Construct – The overall expression of a single set of rules used to apply Context to Core
Components.
Controlled Vocabulary – A supplemental vocabulary used to uniquely define potentially ambiguous
words or Business Terms. This ensures that every word within any of the Core Component names and
definitions is used consistently, unambiguously and accurately.
Core Component (CC) – A building block for the creation of a semantically correct and meaningful
information exchange package. It contains only the information pieces necessary to describe a specific
concept.
Core Component Catalogue – The temporary collection of all metadata about each Core Component
discovered during the development and initial testing of this Core Component Technical Specification,
pending the establishment of a permanent Registry/repository.
Core Component Dictionary – An extract from the Core Component Catalogue that provides a ready
reference of the Core Component through its Dictionary Entry Name, component parts, and definition.
Core Component Library – The Core Component Library is the part of the registry/repository in which
Core Components shall be stored as Registry Classes. The Core Component Library will contain all the
Core Component Types, Basic Core Components, Aggregate Core Components, Basic Business
Information Entities and Aggregate Business Information Entities.
Core Component Property – A business characteristic belonging to the Object Class represented by an
Aggregate Core Component.
Core Component Type (CCT) – A Core Component, which consists of one and only one Content
Component, that carries the actual content plus one or more Supplementary Components giving an
essential extra definition to the Content Component. Core Component Types do not have Business
Semantics.
Data Type – Defines the set of valid values that can be used for a particular Basic Core Component
Property or Basic Business Information Entity Property. It is defined by specifying restrictions on the Core
Component Type that forms the basis of the Data Type.
Definition – This is the unique semantic meaning of a Core Component, Business Information Entity,
Business Context or Data Type.
Dictionary Entry Name – This is the unique official name of a Core Component, Business Information
Entity, Business Context or Data Type in the dictionary.
Geopolitical Context – Geographic factors that influence Business Semantics (e.g., the structure of an
address).
Industry Classification Context – Semantic influences related to the industry or industries of the trading
partners (e.g., product identification schemes used in different industries).
Information Entity – A reusable semantic building block for the exchange of business-related
information.
Lower-Camel-Case (LCC) – a style that capitalizes the first character of each word except the first word and compounds the name.

Naming Convention – The set of rules that together comprise how the Dictionary entry Name for Core Components (See Section 6.1.4.1.4) and Business Information Entities (See Section 6.1.4.2.4) are constructed.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- UN/CEFACT Core Components Technical Specification, Part 8 of the ebXML Framework Version 2.0
- ebXML Technical Architecture Specification v1.04
- OASIS/ebXML Registry Information Model v2.0
- ebXML Requirements Specification v1.06
- Information Technology - Metadata registries: Registry Metamodel, International Standardization Organization, ISO 11179-3
Appendix F. Core Component Schema Module

<?xml version="1.0" encoding="UTF-8"?>
<!--=============================-->
<!-- CCT Core Component Types Schema Module =====-->
<!--=============================-->

Module of Core Component Types,
Agency: UN/CEFACT
VersionID: 1.0
Last change: 02. August 2004
Copyright (C) UN/CEFACT (2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to UN/CEFACT, except as needed for the purpose of developing UN/CEFACT specifications, in which case the procedures for copyrights defined in the UN/CEFACT 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 UN/CEFACT or its successors or assigns.

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

<xsd:schema targetNamespace="urn:un:unece:uncefact:data:draft:CoreComponentTypesSchemaModule:1.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:cct="urn:un:unece:uncefact:data:draft:CoreComponentTypesSchemaModule:1.0"
attributeFormDefault="unqualified">
<!-- Type Definitions =====-->
<xsd:complexType name="AmountType">
<xsd:annotation>
<xsd:documentation xml:lang="en">
  A number of monetary units specified in a currency where the unit of the currency is explicit or implied.
</xsd:documentation>
</xsd:annotation>
<xsd:simpleContent>
  <xsd:extension base="xsd:decimal">
    <xsd:attribute name="amountCurrencyID" type="xsd:token" use="optional">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          UNDT000001</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
than in a separately defined BBIE. </xsd:UsageRuleText>

</xsd:documentation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>

</xsd:attribute>

</xsd:annotation>
<xs:complexType name="CodeType">
  <xs:documentation>The filename of the binary object.</xs:documentation>
  <xs:attribute name="Filname" type="xsd:anyURI" use="optional"/>
  <xs:attribute name="BinaryObjectURI" type="xsd:token" use="optional"/>
  <xs:attribute name="fileExtension" type="xsd:string" use="optional"/>
  <xs:attribute name="CharacterSet" type="xsd:token" use="optional"/>
  <xs:attribute name="Encoding" type="xsd:token" use="optional"/>
</xs:complexType>
<ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:annotation>
</xsd:extensionBase="xsd:token">
</xsd:attribute name="codeListID" type="xsd:token" use="optional">
</xsd:annotation>
</xsd:documentation xml:lang="en">
</ccts:UniqueID>UNDT000007-SC2</ccts:UniqueID>
</ccts:CategoryCode>SC</ccts:CategoryCode>
</ccts:DictionaryEntryName>Code List. Identifier</ccts:DictionaryEntryName>
</ccts:VersionID>1.0</ccts:VersionID>
</ccts:DefinitionText>The identification of a list of codes.</ccts:DefinitionText>
</ccts:PropertyTermName>Identification</ccts:PropertyTermName>
</ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
</ccts:UsageRuleText>Indicates the list of codes from which the code content is drawn.</ccts:UsageRuleText>
</xsd:annotation>
</xsd:attribute name="codeListAgencyID" type="xsd:token" use="optional">
</xsd:annotation>
</xsd:documentation xml:lang="en">
</ccts:UniqueID>UNDT000007-SC3</ccts:UniqueID>
</ccts:CategoryCode>SC</ccts:CategoryCode>
</ccts:DictionaryEntryName>Code List. Agency Identifier</ccts:DictionaryEntryName>
</ccts:VersionID>1.0</ccts:VersionID>
</ccts:DefinitionText>An agency that maintains one or more lists of codes.</ccts:DefinitionText>
</ccts:PropertyTermName>Agency</ccts:PropertyTermName>
</ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
</ccts:UsageRuleText>Use UN EDIFACT 3055 code list.</ccts:UsageRuleText>
</ccts:UsageRuleText>It is recommended that the code list agency is defined either by this identifier or by the code list agency name supplementary component but not both.</ccts:UsageRuleText>
</xsd:documentation>
</xsd:attribute>
</xsd:attribute name="codeListAgencyNameText" type="xsd:token" use="optional">
</xsd:annotation>
</xsd:documentation xml:lang="en">
</ccts:UniqueID>UNDT000007-SC4</ccts:UniqueID>
</ccts:CategoryCode>SC</ccts:CategoryCode>
</ccts:DictionaryEntryName>Code List. Agency Name Text</ccts:DictionaryEntryName>
</ccts:VersionID>1.0</ccts:VersionID>
</ccts:DefinitionText>The name of the agency that maintains the list of codes.</ccts:DefinitionText>
</ccts:CardinalityText>0..1</ccts:CardinalityText>
</ccts:PropertyTermName>Agency Name</ccts:PropertyTermName>
</ccts:RepresentationTermName>Text</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:attribute>
</xsd:attribute name="codeListNameText" type="xsd:token" use="optional">
</xsd:annotation>
</xsd:documentation xml:lang="en">
</ccts:UniqueID>UNDT000007-SC5</ccts:UniqueID>
</ccts:CategoryCode>SC</ccts:CategoryCode>
</ccts:DictionaryEntryName>Code List. Name. Text</ccts:DictionaryEntryName>
</ccts:VersionID>1.0</ccts:VersionID>
</ccts:DefinitionText>The name of a list of codes.</ccts:DefinitionText>
</ccts:CardinalityText>0..1</ccts:CardinalityText>
</ccts:PropertyTermName>Name</ccts:PropertyTermName>
</ccts:RepresentationTermName>Text</ccts:RepresentationTermName>
</xsd:documentation>
</xsd:attribute>
</xsd:attribute name="codeListVersionID" type="xsd:token" use="optional">
</xsd:annotation>
</xsd:documentation xml:lang="en">
</ccts:UniqueID>UNDT000007-SC6</ccts:UniqueID>
The Uniform Resource identifier that identifies where the code list scheme is located.<ccts:DefinitionText>The Uniform Resource Identifier that identifies where the code list is located.</ccts:DefinitionText>

<codeListUniformResource>UNDT000007-SC7</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC6</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC5</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC4</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC3</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC2</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC1</codeListUniformResource>
<codeListUniformResource>UNDT000007-SC0</codeListUniformResource>
<codeListCodeListScheme>UNDT000007-SC10</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC9</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC8</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC7</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC6</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC5</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC4</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC3</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC2</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC1</codeListCodeListScheme>
<codeListCodeListScheme>UNDT000007-SC0</codeListCodeListScheme>

<codeListCodeList>UNDT000007-SC10</codeListCodeList>
<codeListCodeList>UNDT000007-SC9</codeListCodeList>
<codeListCodeList>UNDT000007-SC8</codeListCodeList>
<codeListCodeList>UNDT000007-SC7</codeListCodeList>
<codeListCodeList>UNDT000007-SC6</codeListCodeList>
<codeListCodeList>UNDT000007-SC5</codeListCodeList>
<codeListCodeList>UNDT000007-SC4</codeListCodeList>
<codeListCodeList>UNDT000007-SC3</codeListCodeList>
<codeListCodeList>UNDT000007-SC2</codeListCodeList>
<codeListCodeList>UNDT000007-SC1</codeListCodeList>
<codeListCodeList>UNDT000007-SC0</codeListCodeList>

<codeListNameSpace>UNDT000007-SC10</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC9</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC8</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC7</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC6</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC5</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC4</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC3</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC2</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC1</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC0</codeListNameSpace>

<codeListVersion>1.0</codeListVersion>
<codeListVersion>1.1</codeListVersion>
<codeListVersion>1.2</codeListVersion>
<codeListVersion>1.3</codeListVersion>
<codeListVersion>1.4</codeListVersion>
<codeListVersion>1.5</codeListVersion>
<codeListVersion>1.6</codeListVersion>
<codeListVersion>1.7</codeListVersion>
<codeListVersion>1.8</codeListVersion>
<codeListVersion>1.9</codeListVersion>
<codeListVersion>1.0</codeListVersion>

<codeListVersion>1.0</codeListVersion>
<codeListVersion>1.1</codeListVersion>
<codeListVersion>1.2</codeListVersion>
<codeListVersion>1.3</codeListVersion>
<codeListVersion>1.4</codeListVersion>
<codeListVersion>1.5</codeListVersion>
<codeListVersion>1.6</codeListVersion>
<codeListVersion>1.7</codeListVersion>
<codeListVersion>1.8</codeListVersion>
<codeListVersion>1.9</codeListVersion>
<codeListVersion>1.0</codeListVersion>

<codeListNameSpace>UNDT000007-SC10</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC9</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC8</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC7</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC6</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC5</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC4</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC3</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC2</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC1</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC0</codeListNameSpace>

<codeListVersion>1.0</codeListVersion>
<codeListVersion>1.1</codeListVersion>
<codeListVersion>1.2</codeListVersion>
<codeListVersion>1.3</codeListVersion>
<codeListVersion>1.4</codeListVersion>
<codeListVersion>1.5</codeListVersion>
<codeListVersion>1.6</codeListVersion>
<codeListVersion>1.7</codeListVersion>
<codeListVersion>1.8</codeListVersion>
<codeListVersion>1.9</codeListVersion>
<codeListVersion>1.0</codeListVersion>

<codeListNameSpace>UNDT000007-SC10</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC9</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC8</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC7</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC6</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC5</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC4</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC3</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC2</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC1</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC0</codeListNameSpace>

<codeListVersion>1.0</codeListVersion>
<codeListVersion>1.1</codeListVersion>
<codeListVersion>1.2</codeListVersion>
<codeListVersion>1.3</codeListVersion>
<codeListVersion>1.4</codeListVersion>
<codeListVersion>1.5</codeListVersion>
<codeListVersion>1.6</codeListVersion>
<codeListVersion>1.7</codeListVersion>
<codeListVersion>1.8</codeListVersion>
<codeListVersion>1.9</codeListVersion>
<codeListVersion>1.0</codeListVersion>

<codeListNameSpace>UNDT000007-SC10</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC9</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC8</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC7</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC6</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC5</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC4</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC3</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC2</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC1</codeListNameSpace>
<codeListNameSpace>UNDT000007-SC0</codeListNameSpace>
The format of the date time content should be available across the world wide web.

Use one of the following 8601 format strings:

- CC
- CCYY
- CCYYMM
- MM
- DD
- WW
- MMDD
- hh
- hhmm
- hhmmss

May be used to identify the location of the schema of the code list if information is available across the world wide web.

This is a string formatted to the date format text SC.

If this SC is not used it is assumed that the format is CCYYMMDD; CCYYMMDDhmmssZhhmm; CCYYMMDDhhmmssZhhmm; CCYYMMDDmm; CCYYMMDDZhhmm unless the format is included in the syntax implementation of the Date Content Component.
The name of the agency that maintains the identification scheme. 

Agency. <ccts:UsageRuleText>It is recommended that this is identified either by this identifier or by the identification schema name supplementary component not both.</ccts:UsageRuleText>
A list of two mutually exclusive Boolean values that express the only possible states of a Property. The enumeration is defined by the indicator format text SC.

Whether the indicator is numeric, textual or binary.
<ccts:RepresentationTermName>Measure</ccts:RepresentationTermName>

ccts:DefinitionText

The version of the measure unit code list.

ccts:VersionID

The version of the measure unit code list.

ccts:VersionID

It is recommended that the measure unit code is provided in this SC rather than in a separately defined BBIE.

ccts:UsageRuleText

The type of unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.

ccts:DefinitionText

A numeric value determined by measuring an object along with the specified unit of measure.
Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.

Not to be used as facets can be used instead.
is defined either by identifier or name. The identification of the agency that maintains the quantity unit code

The name of the agency which maintains the quantity unit code

A character string (i.e. a finite set of characters) generally in the form of words of a

language. The identifier of the language used in the content

component.
<ccts:UsageRuleText>Identifies the language in which the code name is presented.</ccts:UsageRuleText>
</xsd:documentation>
</xsd:annotation>
<xsd:attribute name="languageLocaleID" type="xsd:token" use="optional">
<xsd:documentation xml:lang="en">
< ccts:UniqueID>UNDT000019-SC3</ccts:UniqueID>
< ccts:CategoryCode>SC</ccts:CategoryCode>
< ccts:DictionaryEntryName>Language. Locale. Identifier</ccts:DictionaryEntryName>
< ccts:VersionID>1.0</ccts:VersionID>
< ccts:DefinitionText>The identification of the locale of the language.</ccts:DefinitionText>
< ccts:CardinalityText>0..1</ccts:CardinalityText>
< ccts:PropertyTermName>Locale</ccts:PropertyTermName>
< ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
< ccts:UsageRuleText>Not recommended for use.</ccts:UsageRuleText>
</xsd:documentation>
</xsd:attribute>
</xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:schema>
Appendix G. Unqualified Data Type Schema Module

<?xml version="1.0" encoding="UTF-8"?>
<!-- UDT Unqualified Data Types Schema Module -->
<!-- \--------------------------------------------------------------- -->
Module of Unqualified Data Types,
Agency: UN/CEFACT,
Version: 1.0
Last change: 02. August 2004
Copyright (C) UN/CEFACT (2004). All Rights Reserved.
This document and translations of it may be copied and furnished to others,
and derivative works that comment on or otherwise explain it or assist
in its implementation may be prepared, copied, published and distributed,
in whole or in part, without restriction of any kind, provided that the
above copyright notice and this paragraph are included on all such copies
and derivative works. However, this document itself may not be modified in
any way, such as by removing the copyright notice or references to
UN/CEFACT, except as needed for the purpose of developing UN/CEFACT
specifications, in which case the procedures for copyrights defined in the
UN/CEFACT 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 UN/CEFACT or its successors or assigns.
This document and the information contained herein is provided on an "AS IS"
basis and UN/CEFACT DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING
BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL
NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR
FITNESS FOR A PARTICULAR PURPOSE.

<!-- \--------------------------------------------------------------- -->
<xsd:schema targetNamespace="urn:un:unece:uncefact:data:draft:UnqualifiedDataTypesSchemaModule:1:0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:udt="urn:un:unece:uncefact:codelist:draft:UnqualifiedDataTypesSchemaModule:1:0" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <!-- \--------------------------------------------------------------- -->
  <xsd:complexType name="AmountType">
    <xsd:annotation>
      <ccts:UniqueID>UDT000001</ccts:UniqueID>
      <ccts:CategoryCodes>UDT</ccts:CategoryCodes>
      <ccts:DictionaryEntryName>Amount. Type</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>A number of monetary units specified in a currency where the unit of the currency is
explicit or implied.</ccts:DefinitionText>
    </xsd:annotation>
  </xsd:complexType>
</xsd:schema>
<xs:complexType name="Graphic.Type">
  <xs:simpleContent>
    <xs:documentation>The character set of the binary object if the mime type is text.</xs:documentation>
  </xs:simpleContent>
  <xs:extension basisType="UDT000002-SC6" name="Graphic.Type">
    <xs:attribute name="BinaryObjectEncoding.Code">
      <xs:documentation>The Uniform Resource Identifier that identifies where the binary object is located. For example, file://localhost/... or http://... </xs:documentation>
      <xs:annotation>
        <xs:documentation xml:lang="en">The uniform resource identifier that identifies where the binary object is located. For example, file://localhost/... or http://... </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <!-- Secondary RT: Graphic. Type -->
    <!-- Secondary RT: Graphic. Type -->
    <xs:attribute name="BinaryObjectCharacterSetCode">
      <xs:documentation>The Uniform Resource Identifier that identifies where the binary object is located. For example, file://localhost/... or http://... </xs:documentation>
      <xs:annotation>
        <xs:documentation xml:lang="en">The uniform resource identifier that identifies where the binary object is located. For example, file://localhost/... or http://... </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="BinaryObjectURI">
      <xs:annotation>
        <xs:documentation xml:lang="en">The Uniform Resource Identifier that identifies where the binary object is located. For example, file://localhost/... or http://... </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:extension>
</xs:complexType>
<xsd:simpleContent>
  <xsd:extension base="xsd:base64Binary">
    <xsd:attribute name="graphicFormatText" type="xsd:token" use="optional">
      <xsd:documentation xml:lang="en">
        The Uniform Resource Identifier that identifies where the binary object is located.
      </xsd:documentation>
    </xsd:attribute>
    <xsd:attribute name="graphicMimeCode" type="clmIANAMEMediaTypes:BinaryObjectMimeCodeContent_type" use="required">
      <xsd:documentation xml:lang="en">
        Specifies the decoding algorithm of the binary object.
      </xsd:documentation>
    </xsd:attribute>
    <xsd:attribute name="graphicEncodingCode" type="xsd:token" use="optional">
      <xsd:documentation xml:lang="en">
        The format of the binary content.
      </xsd:documentation>
    </xsd:attribute>
    <xsd:attribute name="graphicURI" type="xsd:anyURI" use="optional">
      <xsd:documentation xml:lang="en">
        The Uniform Resource Identifier that identifies where the binary object is located.
      </xsd:documentation>
    </xsd:attribute>
    <xsd:attribute name="graphicFilenameText" type="xsd:token" use="optional">
      <xsd:documentation xml:lang="en">
        The filename of the binary object.
      </xsd:documentation>
    </xsd:attribute>
  </xsd:extension>
</xsd:simpleContent>
<ccts:DefinitionText>The Uniform Resource Identifier that identifies where the binary object is located.</ccts:DefinitionText>

<ccts:CardinalityText>0..1</ccts:CardinalityText>

<ccts:PropertyTermName>Uniform Resource</ccts:PropertyTermName>

<ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>

<ccts:DocumentationText xml:lang="en"/>

<xsd:complexType name="PictureFilenameType">
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
      A diagram, graph, mathematical curves, or similar representation.
    </xsd:documentation>
    <xsd:attribute name="pictureFilenameText" type="xsd:token" use="optional"/>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="SoundType">
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
      Secondary RT: Sound. Type
    </xsd:documentation>
    <xsd:extension base="xsd:64Binary">
      <xsd:attribute name="soundEncodingCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="required"/>
      <xsd:attribute name="soundFormatText" type="xsd:token" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="SoundFormatType">
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
      A diagram, graph, mathematical curves, or similar representation.
    </xsd:documentation>
    <xsd:attribute name="soundFormatText" type="xsd:token" use="optional"/>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="SoundEncodingType">
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
    </xsd:documentation>
    <xsd:attribute name="soundEncodingCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="required"/>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="SoundMimeType">
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
    </xsd:documentation>
    <xsd:attribute name="soundMimeCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="required"/>
    
    <xsd:attribute name="soundEncodingCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="optional"/>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="SoundMimeCode">
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
    </xsd:documentation>
    <xsd:attribute name="soundMimeCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="required"/>
    
    <xsd:attribute name="soundEncodingCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="optional"/>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:documentation xml:lang="en">
  <ccts:UniqueID>UDT000005-SC4</ccts:UniqueID>
  <ccts:CategoryCode>SC</ccts:CategoryCode>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText>Specifies the decoding algorithm of the binary object.</ccts:DefinitionText>
</xsd:documentation>

<xsd:complexType>
  <xsd:simpleContent>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UDT000005-SC6</ccts:UniqueID>
      <ccts:CategoryCode>SC</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Sound. Uniform Resource. Identifier</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>The Uniform Resource Identifier that identifies where the binary object is located.</ccts:DefinitionText>
    </xsd:documentation>
    <xsd:attribute name="soundURI" type="xsd:anyURI" use="optional">
      <xsd:documentation xml:lang="en">
        <ccts:UniqueID>UDT000005-SC8</ccts:UniqueID>
        <ccts:CategoryCode>SC</ccts:CategoryCode>
        <ccts:DictionaryEntryName>Sound. Filename. Text</ccts:DictionaryEntryName>
        <ccts:VersionID>1.0</ccts:VersionID>
        <ccts:DefinitionText>The filename of the binary object.</ccts:DefinitionText>
      </xsd:documentation>
    </xsd:attribute>
  </xsd:simpleContent>
  <xsd:complexType>
    <xsd:attribute name="soundFilenameText" type="xsd:token" use="optional">
      <xsd:documentation xml:lang="en">
        <ccts:UniqueID>UDT000005-SC10</ccts:UniqueID>
        <ccts:CategoryCode>SC</ccts:CategoryCode>
        <ccts:DictionaryEntryName>Sound. Type</ccts:DictionaryEntryName>
        <ccts:VersionID>1.0</ccts:VersionID>
        <ccts:DefinitionText>The format of the binary content.</ccts:DefinitionText>
      </xsd:documentation>
    </xsd:attribute>
  </xsd:complexType>
</xsd:complexType>

<xsd:complexType name="VideoType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      <ccts:UniqueID>UDT000005-SC12</ccts:UniqueID>
      <ccts:CategoryCode>UTD</ccts:CategoryCode>
      <ccts:DictionaryEntryName>Video. Type</ccts:DictionaryEntryName>
      <ccts:VersionID>1.0</ccts:VersionID>
      <ccts:DefinitionText>A diagram, graph, mathematical curves, or similar representation.</ccts:DefinitionText>
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:attribute name="videoFormatText" type="xsd:token" use="optional">
      <xsd:documentation xml:lang="en">
        <ccts:UniqueID>UDT000005-SC14</ccts:UniqueID>
        <ccts:CategoryCode>SC</ccts:CategoryCode>
        <ccts:DictionaryEntryName>Video. Format. Text</ccts:DictionaryEntryName>
        <ccts:VersionID>1.0</ccts:VersionID>
        <ccts:DefinitionText>The format of the binary content.</ccts:DefinitionText>
      </xsd:documentation>
    </xsd:attribute>
  </xsd:complexType>
</xsd:complexType>
located. i
<xs:documentation xml:lang="en">
  <ccts:UniqueID>UDT000006-SC</ccts:UniqueID>
  <ccts:CategoryCode>SC</ccts:CategoryCode>
  <ccts:VersionID>1.0</ccts:VersionID>
  <ccts:DefinitionText>The mime type of the binary object.</ccts:DefinitionText>
  <ccts:CardinalityText>1</ccts:CardinalityText>
  <ccts:PropertyTermName>Mime</ccts:PropertyTermName>
  <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
</xs:documentation>

<xs:attribute name="videoMimeCode" type="clmIANAMIMEMediaTypes:BinaryObjectMimeCodeContentType" use="required">
  <xs:documentation xml:lang="en">
    <ccts:UniqueID>UDT000006-SC3</ccts:UniqueID>
    <ccts:CategoryCode>SC</ccts:CategoryCode>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText>The mime type of the binary object.</ccts:DefinitionText>
    <ccts:CardinalityText>1</ccts:CardinalityText>
    <ccts:PropertyTermName>Mime</ccts:PropertyTermName>
    <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
  </xs:documentation>
</xs:attribute>

<xs:attribute name="videoEncodingCode" type="xsd:token" use="optional">
  <xs:documentation xml:lang="en">
    <ccts:UniqueID>UDT000006-SC4</ccts:UniqueID>
    <ccts:CategoryCode>SC</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Video. Encoding. Code</ccts:DictionaryEntryName>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText>Specifies the decoding algorithm of the binary object.</ccts:DefinitionText>
    <ccts:CardinalityText>1</ccts:CardinalityText>
    <ccts:PropertyTermName>Encoding</ccts:PropertyTermName>
    <ccts:RepresentationTermName>Code</ccts:RepresentationTermName>
  </xs:documentation>
</xs:attribute>

<xs:attribute name="videoURI" type="xsd:anyURI" use="optional">
  <xs:documentation xml:lang="en">
    <ccts:UniqueID>UDT000006-SC6</ccts:UniqueID>
    <ccts:CategoryCode>SC</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Video. Uniform Resource. Identifier</ccts:DictionaryEntryName>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText>The Uniform Resource Identifier that identifies where the binary object is located.</ccts:DefinitionText>
    <ccts:CardinalityText>0..1</ccts:CardinalityText>
    <ccts:PropertyTermName>Uniform Resource</ccts:PropertyTermName>
    <ccts:RepresentationTermName>Identifier</ccts:RepresentationTermName>
  </xs:documentation>
</xs:attribute>

<xs:attribute name="videoFilenameText" type="xsd:token" use="optional">
  <xs:documentation xml:lang="en">
    <ccts:UniqueID>UDT000006-SC7</ccts:UniqueID>
    <ccts:CategoryCode>SC</ccts:CategoryCode>
    <ccts:DictionaryEntryName>Video. Filename. Text</ccts:DictionaryEntryName>
    <ccts:VersionID>1.0</ccts:VersionID>
    <ccts:DefinitionText>The filename of the binary object.</ccts:DefinitionText>
    <ccts:CardinalityText>0..1</ccts:CardinalityText>
    <ccts:PropertyTermName>Filename</ccts:PropertyTermName>
    <ccts:RepresentationTermName>Text</ccts:RepresentationTermName>
  </xs:documentation>
</xs:attribute>
A character string (letters, figures, or symbols) that for brevity and/or language
independence may be used to represent or replace a definitive value or text of an attribute together with relevant
supplementary information.

The Uniform Resource identifier that identifies where the code list scheme
located.

The Uniform Resource Identifier that identifies where the code list is
located.

It is recommended that the name of a code is provided in this SC rather
than in a separately defined BBIE.

May be used to identify the location of the code list if available across the
world wide web.

The Uniform Resource Identifier that identifies where the code list scheme
is located.
The lexical representation for time is the left truncated lexical representation for date:

```
dateTime: hh:mm:ss
```

with optional following time zone indicator. The lexical representation for time is the left truncated lexical representation for time:

```
time: hh:mm:ss
```

The lexical representation for date and time is the extended format:

```
CCYY-MM-DD
```

The instance of time that occurs every day.

```
named: Day
```

One calendar day according the Gregorian calendar.

```
named: Day
```

This lexical representation is the [ISO 8601] extended format CCYY-MM-DD:

```
CCYY-MM-DD
```

available across the world wide web:

```
May be used to identify the location of the schema of the code list if
```

Type: Identifier

Type: Time

Type: Date

Type: DateTime

Primary RT: Identifier

Secondary RT: Time

Secondary RT: Date

Primary RT: Date Time

===================================================================

===== Primary RT: Identifier. Type

===== Secondary RT: Time. Type

===== Secondary RT: Date. Type

===== Primary RT: Date Time. Type

===================================================================

CCTS: UsageRuleText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID

CCTS: DictionaryEntryName

CCTS: CategoryCode

CCTS: UniqueID

CCTS: UsageRuleText

CCTS: RepresentationTermName

CCTS: DefinitionText

CCTS: VersionID
A character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects in the same scheme together with relevant supplementary information.

The Uniform Resource Identifier that identifies where the identification scheme data is located.

The Uniform Resource Identifier that identifies where the identification scheme is located.

May be used to identify the location of the identification scheme if available across the world wide web.

May be used to identify the location of the identification scheme if available across the world wide web.

A list of two mutually exclusive Boolean values that express the only possible states of a property.

A list of two mutually exclusive Boolean values that express the only possible states of a property.
<table>
<thead>
<tr>
<th>Line</th>
<th>XML Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>5649</td>
<td><code>&lt;ccts:UniqueID&gt;UDT0000013&lt;/ccts:UniqueID&gt;</code></td>
</tr>
<tr>
<td>5650</td>
<td><code>&lt;ccts:CategoryCodes&gt;UDT&lt;/ccts:CategoryCode&gt;</code></td>
</tr>
<tr>
<td>5651</td>
<td><code>&lt;ccts:DictionaryEntryName&gt;Measure. Type&lt;/ccts:DictionaryEntryName&gt;</code></td>
</tr>
<tr>
<td>5652</td>
<td><code>&lt;ccts:VersionID&gt;1.0&lt;/ccts:VersionID&gt;</code></td>
</tr>
<tr>
<td>5653</td>
<td><code>&lt;ccts:DefinitionText&gt;A numeric value determined by measuring an object along with the specified unit of measure.&lt;/ccts:DefinitionText&gt;</code></td>
</tr>
<tr>
<td>5654</td>
<td><code>&lt;ccts:RepresentationTermName&gt;Measure&lt;/ccts:RepresentationTermName&gt;</code></td>
</tr>
<tr>
<td>5655</td>
<td><code>&lt;ccts:PropertyTermName&gt;MeasureUnitCode&lt;/ccts:PropertyTermName&gt;</code></td>
</tr>
<tr>
<td>5656</td>
<td><code>&lt;xsd:complexType&gt;</code></td>
</tr>
<tr>
<td>5657</td>
<td><code>&lt;xsd:annotation&gt;</code></td>
</tr>
<tr>
<td>5658</td>
<td><code>&lt;xsd:simpleContent&gt;</code></td>
</tr>
<tr>
<td>5659</td>
<td><code>&lt;xsd:extension base=&quot;xsd:decimal&quot;&gt;</code></td>
</tr>
<tr>
<td>5660</td>
<td><code>&lt;xsd:attribute name=&quot;measureUnitCode&quot; type=&quot;clm66411:UnitCodeContentType&quot; use=&quot;required&quot;&gt;</code></td>
</tr>
<tr>
<td>5661</td>
<td><code>&lt;xsd:documentation xml:lang=&quot;en&quot;&gt;</code></td>
</tr>
<tr>
<td>5662</td>
<td><code>&lt;ccts:UniqueID&gt;UDT0000013-SC2&lt;/ccts:UniqueID&gt;</code></td>
</tr>
<tr>
<td>5663</td>
<td><code>&lt;ccts:CategoryCode&gt;SC&lt;/ccts:CategoryCode&gt;</code></td>
</tr>
<tr>
<td>5664</td>
<td><code>&lt;ccts:DictionaryEntryName&gt;Measure. Unit. Code&lt;/ccts:DictionaryEntryName&gt;</code></td>
</tr>
<tr>
<td>5665</td>
<td><code>&lt;ccts:VersionID&gt;1.0&lt;/ccts:VersionID&gt;</code></td>
</tr>
<tr>
<td>5666</td>
<td><code>&lt;ccts:DefinitionText&gt;The type of unit of measure.&lt;/ccts:DefinitionText&gt;</code></td>
</tr>
<tr>
<td>5667</td>
<td><code>&lt;ccts:PropertyTermName&gt;MeasureUnit&lt;/ccts:PropertyTermName&gt;</code></td>
</tr>
<tr>
<td>5668</td>
<td><code>&lt;ccts:RepresentationTermName&gt;BaseCode&lt;/ccts:RepresentationTermName&gt;</code></td>
</tr>
<tr>
<td>5669</td>
<td><code>&lt;xsd:complexType&gt;</code></td>
</tr>
<tr>
<td>5670</td>
<td><code>&lt;xsd:annotation&gt;</code></td>
</tr>
<tr>
<td>5671</td>
<td><code>&lt;xsd:extension base=&quot;xsd:decimal&quot;&gt;</code></td>
</tr>
<tr>
<td>5672</td>
<td><code>&lt;xsd:attribute name=&quot;baseMeasureUnit&quot; type=&quot;clm66411:MeasureUnitContentType&quot; use=&quot;required&quot;&gt;</code></td>
</tr>
<tr>
<td>5673</td>
<td><code>&lt;xsd:documentation xml:lang=&quot;en&quot;&gt;</code></td>
</tr>
<tr>
<td>5674</td>
<td><em>A numeric value determined by measuring an object along with the specified unit of measure.</em></td>
</tr>
</tbody>
</table>
A character string (i.e. a finite set of characters) generally in the form of words of a language.

Sequence. It does not require a unit of quantity or unit of measure.

The unit of the quantity

A counted number of non-monetary units possibly including fractions.

It is recommended that if a quantity unit code is provided it should be provided in this SC rather than in a separately defined BBIE.

If no code list is specified it is assumed to be UNECE Rec 20 (Common Codes).

Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.
5791  </xsd:documentation>
5792  </xsd:annotation>
5793  </xsd:simpleContent>
5794  </xsd:extension base="xsd:string">
5795  
5796  <xsd:attribute name="languageID" type="clm5639:LanguageCodeContentType" use="optional">
5797  </xsd:attribute>
5798  </xsd:documentation xml:lang="en">
5799  
5800  </xsd:attribute>
5801  </xsd:documentation xml:lang="en">
5802  </xsd:extension>
5803  
5804  </xsd:extension base="xsd:string">
5805  
5806  </xsd:attribute>
5807  </xsd:documentation>
5808  
5809  </xsd:extension base="xsd:string">
5810  
5811  </xsd:attribute>
5812  </xsd:extension>
5813  
5814  </xsd:extension base="xsd:string">
5815  
5816  </xsd:extension>
5817  
5818  </xsd:attribute>
5819  </xsd:documentation>
5820  
5821  </xsd:attribute>
5822  </xsd:extension>
5823  </xsd:extension base="xsd:string">
5824  
5825  </xsd:extension>
5826  
5827  
5828  </xsd:extension base="xsd:string">
5829  
5830  
5831  
5832  </xsd:extension base="xsd:string">
5833  
5834  </xsd:extension>
5835  
5836  </xsd:attribute>
5837  </xsd:extension base="xsd:string">
5838  
5839  </xsd:extension>
5840  
5841  </xsd:attribute>
5842  </xsd:extension>
5843  </xsd:extension base="xsd:string">
5844  
5845  </xsd:extension>
5846  
5847  </xsd:extension>
5848  
5849  </xsd:extension>
5850  
5851  </xsd:extension>
5852  </xsd:extension>
5853  
5854  </xsd:extension base="xsd:string">
5855  
5856  </xsd:extension>
5857  </xsd:extension>
5858  </xsd:extension>
5859  </xsd:extension>
5860  </xsd:extension>
5861  </xsd:extension>

NamingAndDesignRules_1.0.doc Page 124 8/12/2004
<xsd:attribute name="languageLocaleID" type="xsd:token" use="optional">
    <xsd:documentation xml:lang="en">
        The identification of the locale of the language.
    </xsd:documentation>
    <xsd:annotation>
        <ccts:UniqueID>UDT0000020-SC3</ccts:UniqueID>
        <ccts:CategoryCode>SC</ccts:CategoryCode>
        <ccts:DictionaryEntryName>Language. Locale. Identifier</ccts:DictionaryEntryName>
        <ccts:VersionID>1.0</ccts:VersionID>
        <ccts:DefinitionText>The identification of the locale of the language.</ccts:DefinitionText>
        <ccts:CardinalityText>0..1</ccts:CardinalityText>
        <ccts:UsageRuleText>Not recommended for use.</ccts:UsageRuleText>
    </xsd:annotation>
</xsd:attribute>
Copyright Statement

Copyright © UN/CEFACT 2002. All Rights Reserved.

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

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

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