United Nations Centre for Trade Facilitation and Electronic Business

UN/CEFACT
UML Profile for Core Components (UPCC)

Version 1.0

Final Specification

2008-01-16

Table of Contents

1 This Document

1.1 Context

UN/CEFACT metamodels for business information (CCTS) and business process modelling (UMM) provide conceptual metamodels. For these to be usable with common modelling tools that support UML and to be able to use UML infrastructure for validation the conceptual models need to be mapped to the UML meta-model. This document provides a mapping of the CCTS to UML as a formal UML profile.

1.2 Conventions

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 [RFC2119] as quoted here:

- MUST: This word, or the terms "REQUIRED" or "SHALL", means that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications MUST be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation that does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation that does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

2 Project Team Participants

2.1 Disclaimer

The views and specification expressed in this document are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this specification.

2.2 Contact Information

Project Team Lead	Steve Capell	steve.capell@redwahoo.com
Editor	Jens Dietrich	jens.dietrich@gmx.de
Contributors	Fabian Büttner	green@tzi.de
	Mike Conroy	michael.conroy@wanadoo.fr
	Philipp Liegl	philipp.liegl@gmx.at
	Harry Moyer	harry.moyer@redwahoo.com
	Joern Guy Süss	joern.guy.suess@web.de
	Christian Senf	christian.senf@sysedv.tu-berlin.de
	Anders Tell	anderst@toolsmiths.se
	Jim Wilson	jim.wilson@cidx.org

3 Introduction

3.1 Summary of Contents of Document

3.2 Audience

The target audience for this document includes members of the UN/CEFACT TMG, other UN/CEFACT working groups, the UN/CEFACT Forum management group, and the wider community of modeling tool vendors.

3.3 Related Documents

UN/CEFACT Specifications

UMM Base Module 1.0
http://www.untmg.org/dmdocuments/UMM_Base_Module_V10_Specification_20061006.pdf

UMM Foundation Module 1.0
http://www.untmg.org/dmdocuments/UMM_Foundation_Module_V10_Specification_20061006.pdf

CCTS 2.01
http://www.untmg.org/dmdocuments/CCTS_v201_2003_11_15.pdf

ATG2 XML NDR
http://www.disa.org/cefact-groups/atg/downloads/XML%20Naming%20And%20Design%20Rules%20V2.0.pdf

OMG Specifications

UML 2.0	http://www.omg.org/cgi-bin/doc?formal/05-07-04 http://www.omg.org/cgi-bin/doc?formal/05-07-05
UML 1.4.2 (ISO)	http://www.omg.org/cgi-bin/doc?formal/04-07-02
MOF 1.3	http://www.omg.org/cgi-bin/doc?formal/00-04-03
XMI 1.2	http://www.omg.org/cgi-bin/doc?formal/2002-01-01
XMI 2.1	http://schema.omg.org/spec/XMI/2.1

Other Related Documents

Eclipe eCore	http://www.eclipse.org/emf/2002/Ecore
Eclipse UML2 Project	http://www.eclipse.org/uml2/2.0.0/UML

4 Objectives

4.1 Goals

The business goals of this specification are:

- To make CCTS compliant information modelling accessible to a broad user base through standard UML tool support.
- To support easy interchange of information models between different UML tools.
- To support validation of the structure and semantics of information models against the CCTS.

These goals are achieved through the development of a formal UML profile for CCTS that includes stereotypes, tagged values and OCL constraints.

4.2 Requirements

This specification is guided by the following key requirements:

- That accessibility to a large community through availability of high quality tools is the key objective.
- That the UML profile should minimise complexity for the business user.
- That the UML profile provides a complete representation of data required by CCTS
- The UML profile for core components complements the base and foundation profiles for UMM.
- The UML profile may need to define different levels of compliance – depending on the requirements of the modeler and the capabilities of the tool.
- The UML profile shall be implementable by the widest group of tool vendors.
- The UML profile will support the generation of XML schema from the model that are compliant with ATG2 XML Naming and Design Rules .

4.3 Caveats and Assumptions

This specification makes the following assumptions:

- That the XMI generated by modeling tools will be compliant with the proposed Eclipse UML2 reference implementation based on XMI 2.1 (http://schema.omg.org/spec/XMI/2.1) and UML 2.0 (http://www.eclipse.org/uml2/2.0.0/UML)
- That most modeling tools do not evaluate OCL constraints against model data. Accordingly, validation of CCTS semantics as defined by the OCL constraints in this specification will normally only be possible using either an external validation service or a custom plug-in.
- The UML profile does not specify requirements for diagram interchange.

5 Overview

5.1 The model interchange problem

The OMG XML Metadata Interchange (XMI) specification has been defined to support exchange of model data between different tools. However:

- There are several version of XMI and different tools may support different versions.
- The XMI is just a description of a model instance in terms of the reference UML meta-model. But since there are several version of the UML reference model, files exported from one tool may not be understood by another.
- The objective being to use UML as a notation to describe information and process models, the description of entities such as “BCC”, “ABIE”, “Transactions”, “roles”, etc is not uniform between different tools since they are not UML concepts. So one modeler might choose to represent a BBIE in one way while a different modeler will choose to represent the same BIE using a different UML model element. Consequently models will not be re-usable amongst different users and the goal of generating deployment schema like ATG2 XSD, BPSS or BPEL from models cannot be realized.

Consequently, to obtain a consistant interchange at the Model level it is necessary to:

- Specify the supported UML versions that will be used. This specification supports UML 1.4.2 (which is also the ISO standard version) and UML2.0.
- Define precise mappings from CCTS / UMM concepts to corresponding UML elements and to provide these as a “UML profile” that consists of stereotypes, tagged values and OCL constraints. That is the purpose of this specification.
- Propose an XMI implementation as a reference that ensures consistent model exchange.

This specification shall therefore deliver:

- A normative specification (this document) that defines the UML profile for CCTS as a set of model diagrams showing stereotypes and tagged values plus a set of OCL constraints that describe the semantic restrictions imposed by CCTS.
- An XMI file containing the UML profile for Core Components that can be imported into any compliant modeling tool.

5.2 The UML profile approach

The approach taken by this specification to deliver on the goal of interchange of structurally and semantically valid information models is shown in the diagram below.

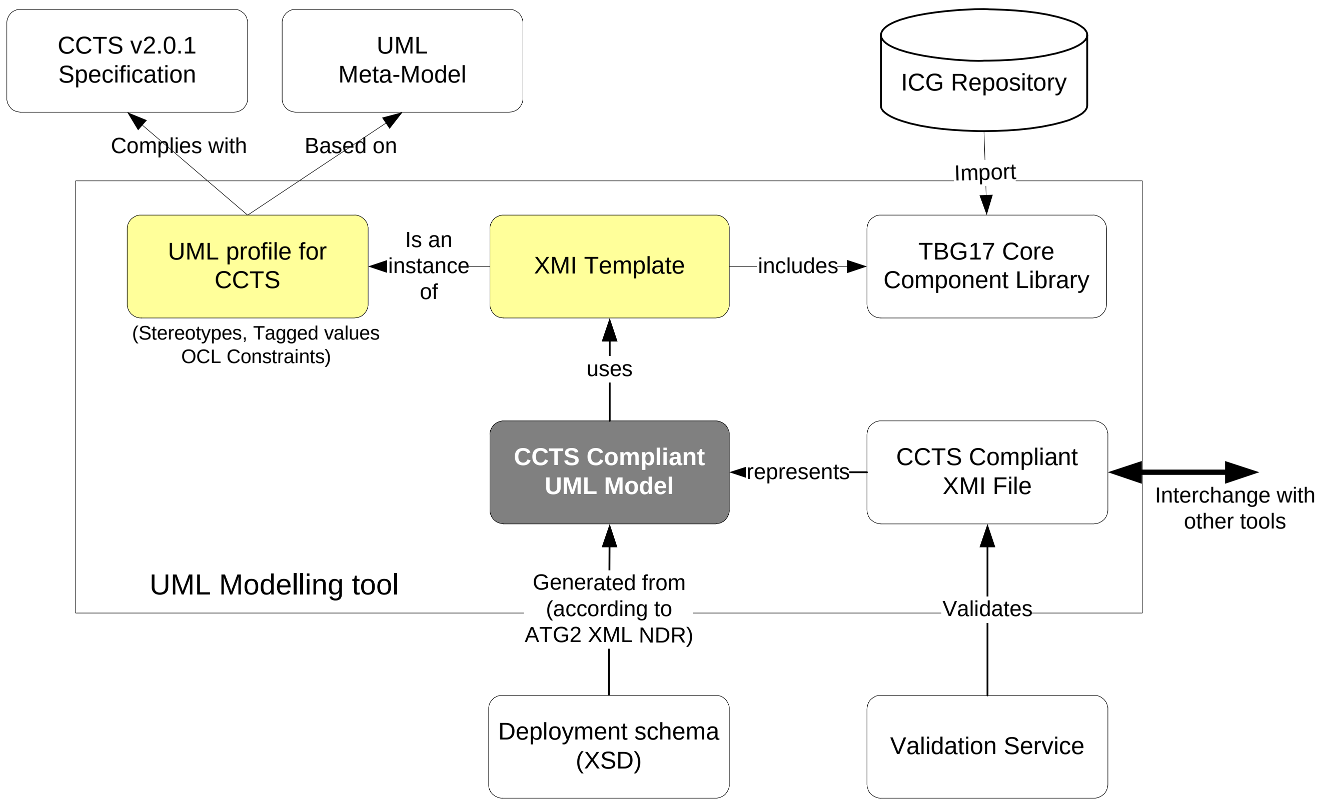


Figure 2 – UML Profile Usage

- The CCTS v2.01 is the reference for the structural and semantic constraints that the UML profile should reflect.
- The UML profile is provided as an XMI Template file in the recommended XMI-interchange format (EMF UML2 XMI). .
- Modellers shall use the template together with library components (that also conform to this profile) to build their CCTS compliant information models.
- An interoperable XMI representation of the model can be generated from tools that support the XMI-Format accepted by the Eclipse UML2 reference implementation based on XMI 2.1 (http://schema.omg.org/spec/XMI/2.1) and UML 2.0 (http://www.eclipse.org/uml2/2.0.0/UML) This Format will be referred to as “EMF UML2 XMI”. The XMI can be sent to a validation service, submitted to a repository, or exchanged with other modellers.
- ATG2 XML NDR compliant deployment schema can be generated directly from the model.

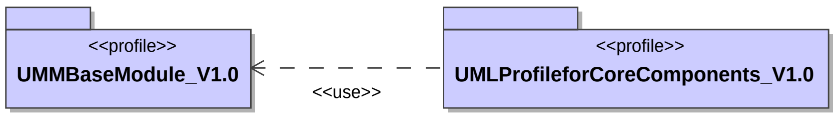
6 Core Component Module

This section is *normative.* This section provides the detailed definition of the UML profile for Core Components.

6.1 Conceptual Metamodel

The conceptual meta-model to which this profile must comply is defined in the Core Component Technical Specification Version 2.1 and is not reproduced here.

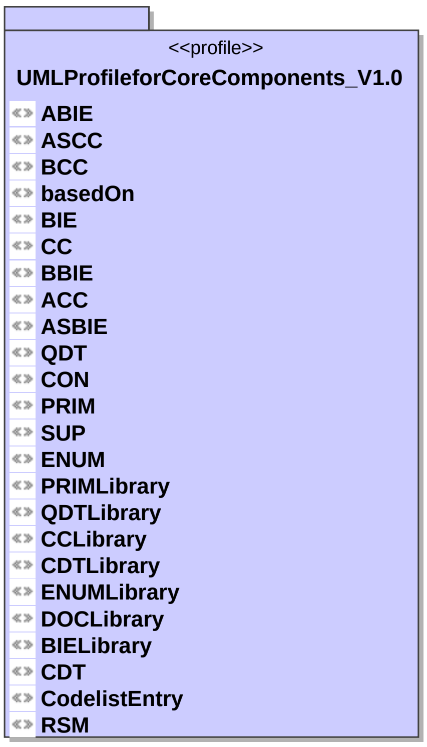
6.2 Relationships and dependencies



The CCTS module uses the RegistryObject and BusinessLibraryPackage elements from the UMM Base Module.

6.3 Overview

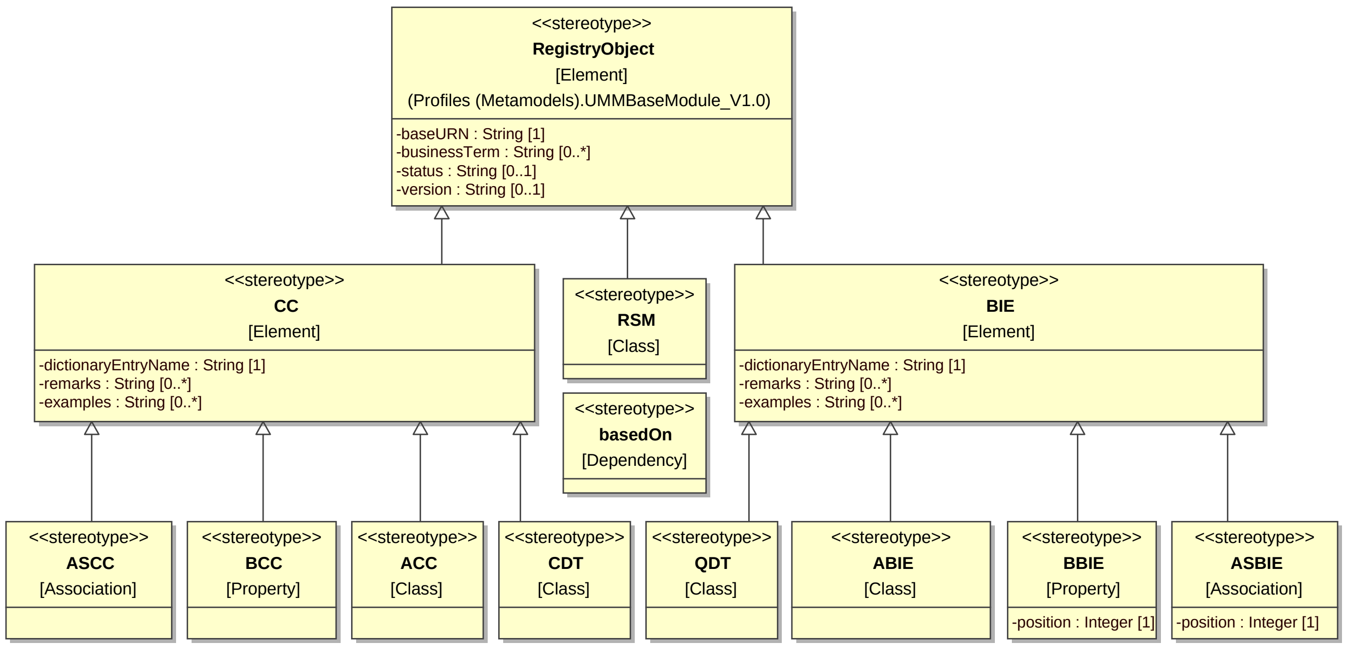
The below figure gives an overwiev of all Stereotypes of the UPPC. The referred acronyms are described later in this chapter .



6.4 Common Stereotypes

6.4.1 Stereotypes and Tag Definitions (normative)

The following diagram shows a visualization of the Common Stereotypes to provide an overview of the defined elements. Note that the diagram shows only the structure of the UML Profile. Additional constraints to define compliance with the semantics of the CCTS meta-model are provided in a subsequent section.



<<stereotype>> RegistryObject [Element] (Profiles (Metamodels).UMMBaseModule_V1.0)
-baseURN : String [1]
-businessTerm : String [0..*]
-status : String [0..1]
-version : String [0..1]
<<stereotype>> CC [Element]
-dictionaryEntryName : String [1]
-remarks : String [0..*]
-examples : String [0..*]
<<stereotype>> RSM [Class]
<<stereotype>> basedOn [Dependency]
<<stereotype>> BIE [Element]
-dictionaryEntryName : String [1]
-remarks : String [0..*]
-examples : String [0..*]
<<stereotype>> ASCC [Association]
<<stereotype>> BCC [Property]
<<stereotype>> ACC [Class]
<<stereotype>> CDT [Class]
<<stereotype>> QDT [Class]
<<stereotype>> ABIE [Class]
<<stereotype>> BBIE [Property]
-position : Integer [1]
<<stereotype>> ASBIE [Association]
-position : Integer [1]

The sections below provide details on each stereotype of the Common Stereotypes. Both abstract and concrete stereotypes are described. However OCL constraints are only defined for concrete elements because these are the only ones that will exist in an actual instance model.

6.4.1.1 CC

Stereotype	CC (Abstract)			
Base Class	Element (from UML 1.4.2 / 2.0 meta-model)			
Parent	RegistryObject (from UMM base module)			
Description	The CC (Core Component) element is a superclass to provide attributes common to BCC, ASCC, ACC, and CDT types.			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	dictionaryEntryName	string	1:1	Official Name of the Core Component.
	remarks	string	0:1	Additional remarks to describe the CC.
	examples	string	0:1	Examples to describe the CC.

6.4.1.2 BIE

Stereotype	BIE (Abstract)
Base Class	Element (from UML 1.4.2 / 2.0 meta-model)
Parent	RegistryObject (from UMM base module)
Description	The BIE (Business Information Entity) element is a superclass to provide attributes common to BBIE, ASBIE, ABIE and QDT types.
Tag Definitions	

Tag Definition	**Type**	**Multi-plicity**	**Description**
dictionaryEntryName	string	1:1	Official Name of the Core Component
remarks	string	0:1	Additional remarks to describe the BIE.
examples	string	0:1	Examples to describe the CC.

6.4.1.3 BCC

Stereotype	BCC
Base Class	Attribute (from UML 1.4.2 / 2.0 meta-model)
Parent	CC
Description	The BCC (Basic Core Component) is a singular business characteristic of an ACC (Aggregate Core Component).
Tag Definitions	

Tag Definition	**Type**	**Multi-plicity**	**Description**
N/A			

6.4.1.4 ACC

Stereotype	ACC
Base Class	Class (from UML 1.4.2 / 2.0 meta-model)
Parent	CC
Description	The ACC (Aggregate Core Component) is a collection of related business information that together carry a distinct meaning.
Tag Definitions	

Tag Definition	**Type**	**Multi-plicity**	**Description**
N/A			

6.4.1.5 ASCC

Stereotype	ASCC
Base Class	Association (from UML 1.4.2 / 2.0 meta-model)
Parent	CC
Description	The ASCC (Association Core Component) represents a complex characteristic of an associated ACC. The ACC describes the structure of the ASCC.
Tag Definitions	

Tag Definition	**Type**	**Multi-plicity**	**Description**

	Tag Definition	Type	Multi-plicity	Description
	N/A			

6.4.1.6 BBIE

Stereotype	BBIE			
Base Class	Attribute (from UML 1.4.2 / 2.0 meta-model)			
Parent	BIE			
Description	The BBIE (Basic Business Information Entity) is a singular business characteristic of an ABIE in a specific business context. It is related to the BCC from which it is derived and is linked to a Data Type that describes its values.			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	position	integer	1:1	The position of the BBIE element within an ABIE and platform specific artefacts (e.g. XML schema) generated from the model.

6.4.1.7 ABIE

Stereotype	ABIE			
Base Class	Class (from UML 1.4.2 / 2.0 meta-model)			
Parent	BIE			
Description	The ABIE (Aggregate Business Information Entity) is a collection of related business information that together express a business meaning in a specific context. An ABIE is related to the ACC from which it is derived.			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	N/A			

6.4.1.8 RSM

Stereotype	RSM			
Base Class	Class (from UML 1.4.2 / 2.0 meta-model)			
Parent	RegistryObject (from UMM Base Module)			
Description	An RSM is a Rootschema Module. A Class stereotyped with RSM becomes a root element in platform specific artefacts (like XML schema) generated from the model.			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	N/A			

6.4.1.9 ASBIE

Stereotype	ASBIE
Base Class	Association (from UML 1.4.2 / 2.0 meta-model)
Parent	BIE
Description	The ASBIE (Association Business Information Entity) represents a complex business characteristic of

	an associated ABIE that describes it's structure. An ASBIE is related to the ASCC from which it is derived.			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	position	integer	1:1	The position of the ASBIE element within an ABIE and platform specific artefacts (e.g. XML schema) generated from the model.

6.4.1.10 *basedOn*

Stereotype	basedOn			
Base Class	Dependency (from UML 1.4.2 / 2.0 meta-model)			
Parent	N/A			
Description	The basedOn stereotype is used to define the relationships between • A QDT and the CDT upon which it is based • An ABIE and the ACC upon which it is based			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	fractionDigits	string	0:1	The tags can be used to restrict the content component of a QDT based on a CDT (e.g. a QDT based on CDT "Text" can be restricted to a length of 4).
	length	string	0:1	
	maxExclusive	string	0:1	
	maxInclusive	string	0:1	
	maxLength	string	0:1	
	minExclusive	string	0:1	
	minInclusive	string	0:1	
	minLength	string	0:1	
	pattern	string	0:1	
	totalDigits	string	0:1	
	whiteSpace	string	0:1	

6.4.2 Constraints (normative)

A BCC type must be typed with a class of stereotype <<CDT>>.

UML 1.4
```
context Attribute
      inv BCC_CDT:
            self.isBCC() implies
            self.type.oclAsType(ModelElement).isCDT()
```

UML 2.0
```
context Property
      inv BCC_CDT:
            self.isBCC() implies self.type.isCDT()
```

An ACC must contain only BCCs and ASCCs. There must be at least one BCC or ASCC.

UML 1.4
```
context Class
      inv ACC_AtLeastOneBCCorASCCandNothingElse:
            self.isACC() implies
            self.ownedAttribute-&gt;exists(a |
            a.isBCC() or (a.association.isDefined and a.association.isASCC()))
            and self.ownedAttribute-&gt;forAll(a | a.isBCC()
            (a.association.isDefined and a.association.isASCC()))
```

UML 2.0
```
context Class
      inv ACC_AtLeastOneBCCorASCCandNothingElse:
            self.isACC() implies
            self.ownedAttribute-&gt;exists(a |
            a.isBCC() or (a.association.isDefined and a.association.isASCC()))
            and self.ownedAttribute-&gt;forAll(a | a.isBCC()
            (a.association.isDefined and a.association.isASCC()))
```

A BBIE type must be typed with a class of stereotype either <<QDT>> or <<CDT>>.

UML 1.4
```
context Attribute
      inv BBIE_QDT:
            self.isBBIE() implies
            self.type.oclAsType(ModelElement).isCDT() or
            self.type.oclAsType(ModelElement).isQDT()
```

UML 2.0
```
context Property
      inv BBIE_QDT:
            self.isBBIE() implies (self.type.isCDT() or self.type.isQDT())
```

An ABIE must contain only BBIEss and ASBIEs. There must be at least one BBIE or ASBIE.

UML 1.4
```
context Class
      inv ABIE_AtLeastOneBBIEorASBIEandNothingElse:
            self.isABIE() implies
            self.ownedAttribute-&gt;exists(a |
            a.isBBIE() or (a.association.isDefined and a.association.isASBIE()))
            and self.ownedAttribute-&gt;forAll(a | a.isBBIE()
            (a.association.isDefined and a.association.isASBIE()))
```

UML 2.0
```
context Class
      inv ABIE_AtLeastOneBBIEorASBIEandNothingElse:
            self.isABIE() implies
            self.ownedAttribute-&gt;exists(a |
            a.isBBIE() or (a.association.isDefined and a.association.isASBIE()))
            and self.ownedAttribute-&gt;forAll(a | a.isBBIE()
            (a.association.isDefined and a.association.isASBIE()))
```

An ABIE must have a dependancy of stereotype <<basedOn>> with an ACC as the target.

UML 1.4
```
context Class
      inv ABIE_basedOnACC:
            self.isABIE() implies
            self.supplierDependency->one(isBasedOnToACC())
```

UML 2.0
```
context NamedElement
      inv ABIE_basedOnACC:
            self.isABIE() implies self.clientDependency-&gt;exists(isBasedOnToACC())
```

The originating association end of a stereotype ASCC must be set to AggregationKind::composite

UML 1.4
```
context Association
      inv ASCCAssociationEnd_composite:
            self.isASCC() implies
            self.isComposition()
```

UML 2.0
```
context Class
      inv ASCCAssociationEnd_composite:
            self.isASCC() implies
            self.memberEnd-&gt;exists(e|e.aggregation=#composite)
```

For each ACC/ABIE, all navigable properties (attributes and association ends) must have unique names.

UML 1.4
```
context Class
      inv propertiesOfACCOrABIEHaveUniqueNames:
            self.isACC() or self.isABIE() implies
            self.attribute-&gt;forAll( p1,p2 | p1.name = p2.name implies p1=p2)
```

UML 2.0
```
context Class
      inv propertiesOfACCOrABIEHaveUniqueNames:
            self.isACC() or self.isABIE() implies
            self.attribute-&gt;forAll( p1,p2 | p1.name = p2.name implies p1=p2)
```

6.4.3 Common Stereotypes – Mapping to CCTS

CCTS Construct	UML Implementation
Parent objects - properties inherited by CDT, QDT, ACC, BCC, ASCC, ABIE, BBIE, ASBIE	
Registry Class - Definition	Implemented using the UML 1.4.2 “Comment” element. UML modelling tools normally implement the “Comment” element as a notes area on data entry window for all model elements. Accordingly business users may find it simpler to use the notes field for CCTS definitions.
Registry Class – Dictionary EntryName	Tagged value with name = “dictionaryEntryName”. Implemented at CC/BIE level because registry object is re-used by UMM Foundation. Note that no extra information is introduced by this tagged value because the dictionary entry name is already derivable from class, attribute, and data type names. Business users may prefer to leave this value out and then apply a transform to the resulting XMI.
Core Component Elements	
Aggregate Core Component – Object Class Term	Name of a UML class with stereotype <<ACC>>. Note that this is just the object class term and should not include the “.Details” suffix.
Basic Core Component – property term	Name of an UML attribute with stereotype <<BCC>>. Note that this is just the property term and should not include the object class or representation terms. Special Note: in the rare case where an ACC has two BCCs with the same property term then the representation term should be added to the UML attribute name in order to ensure uniqueness of UML attributes in a class.
Basic Core Component - cardinality	UML cardinality of an attribute with stereotype <<BCC>>.
Basic Core Component – representation term	UML type of an attribute with stereotype <<BCC>>. The type of the BCC must be another class of stereotype <<CDT>> (Core Data Type). Note that the representation term is equal to the name of the CDT class.
Association Core Component – property term	Target role name of an aggregation of stereotype <<ASCC>>. Note that the name is just the property term and not the dictionary entry name.

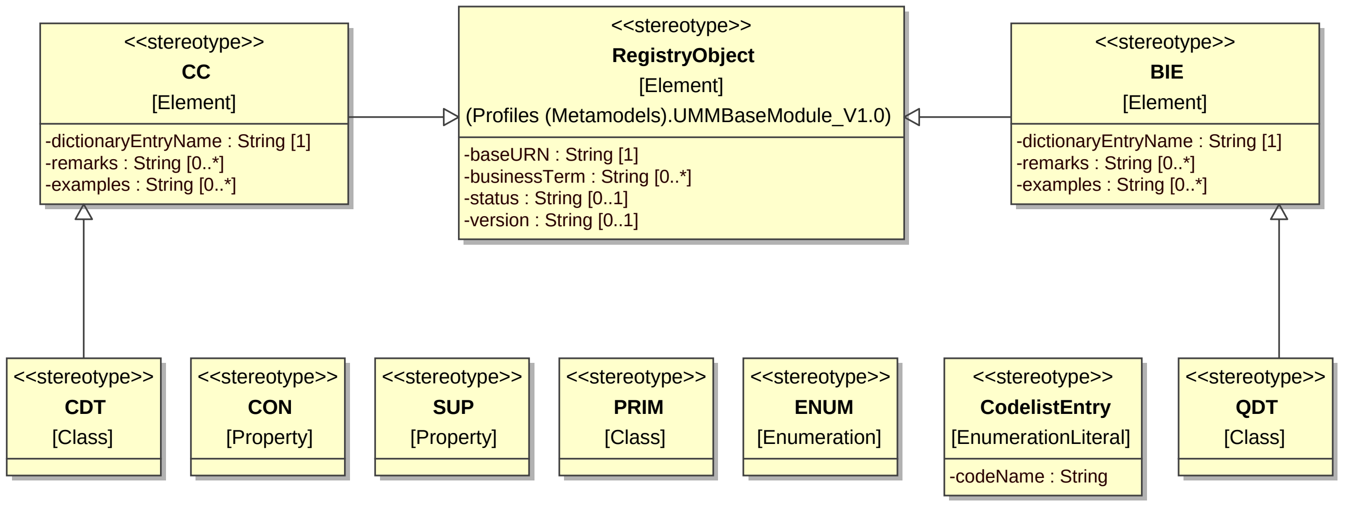
Business Information Entity Elements	
ABIE / BBIE / ASBIE / QDT Qualifier Terms	The qualifier prefix (characters before the underscore) of the ABIE, BBIE, ASBIE and QDT name.
Aggregate Business Information Entity – Object class term	Name of a UML class with stereotype <<ABIE>> (without qualifier prefix if it exists) Note: The ABIE class name is the qualified object class term - shall be equal to the object class name of the ACC upon which this ABIE is based with an optional prefix equal to the qualifier term.
Basic Business Information Entity – property term	Name of a UML attribute with stereotype <<BBIE>> (without qualifier prefix if it exists) Note: The BBIE attribute name is the qualified property term - shall be equal to the property name of the BCC upon which this BBIE is based with an optional prefix equal to the qualifier term. Special Note: in the rare case where an ABIE has two BBIEs with the same property term then the representation term should be added to the UML attribute name in order to ensure uniqueness of UML attributes in a class.
Basic Business Information Entity – representation term	UML type of an attribute with stereotype <<BBIE>>. The type of the BBIE must be another class of stereotype <<CDT>> (Core Data Type) or <<QDT>> (Qualified Data Type). Note that the representation term is equal to the name of the CDT class (or QDT class without qualifier term)
Association Business Information Entity – property term	UML aggregation source role name of an aggregation of stereotype <<ASBIE>>. Note that the name is just the property term and not the dictionary entry name.
“basis” relationship between ABIE and ACC	UML Dependency of stereotype <<basedOn>> between ABIE and ACC with target ACC.
“basis” relationship between BBIE and BCC	No explicit association. BBIE property terms are the same as BCC property terms and are unique within the aggregate. Therefore the explicit dependency between ABIE and ACC is sufficient to infer this relationship. Practical note: in most modeling tools a dependency between attributes is not supported. Workarounds are generally inconvenient to the modeler.
“basis” relationship	No explicit association.

between ASBIE and ASCC	ASBIE property terms are the same as ASCC property terms and are unique within the aggregate. Therefore the explicit dependency between ABIE and ACC is sufficient to infer this relationship

6.5 Data Types Stereotypes

6.5.1 Stereotypes and Tag Definitions (normative)

The following diagram shows a visualization of the Data Types Stereotypes to provide an overview of the defined elements. Note that the diagram shows only the structure of the UML Profile. Additional constraints to define compliance with the semantics of the CCTS meta-model are provided in a subsequent section.



The sections below provide details on each stereotype of the Data Types Stereotypes. Both abstract and concrete stereotypes are described. However OCL constraints are only defined for concrete elements because these are the only ones that will exist in an actual instance model.

6.5.1.1 CDT

Stereotype	CDT
Base Class	Class (from UML 1.4.2 / 2.0 meta-model)
Parent	CC
Description	The CDT (Core Data Type) element is a complex data type that must be one of the approved Core Component Types listed in the CCTS v 2.0.1 section 8.
Tag Definitions	

Tag Definition	Type	Multi-plicity	Description
N/A			

6.5.1.2 QDT

Stereotype	QDT
Base Class	Class (from UML 1.4.2 / 2.0 meta-model)
Parent	BIE
Description	The QDT (Qualified Data Type) element is a complex data type. It is a restriction of a corresponding CDT (Core Data Type) element.

Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	N/A			

6.5.1.3 CON

Stereotype	CON
Base Class	Attribute (from UML 1.4.2 / 2.0 meta-model)
Parent	N/A
Description	The CON (Content Component) element carries the actual content. The type of the CON element shall be a class of stereotype <<PRIM>>.

Tag Definitions	**Tag Definition**	**Type**	**Multipli city**	**Description**
	N/A			

6.5.1.4 SUP

Stereotype	SUP
Base Class	Attribute (from UML 1.4.2 / 2.0 meta-model)
Parent	N/A
Description	The SUP (Supplementary Component) element provides additional meaning to the content component of a CDT or QDT.

Tag Definitions	**Tag Definition**	**Type**	**Multipli city**	**Description**
	N/A			

6.5.1.5 PRIM

Stereotype	PRIM
Base Class	Class (from UML 1.4.2 / 2.0 meta-model)
Parent	N/A
Description	The PRIM (Primitive Type) represents one of the CCTS defined primitive types. All CON and SUP typs must be drawn from the collection of objects of stereotype PRIM.

Tag Definitions	**Tag Definition**	**Type**	**Multipli city**	**Description**
	N/A			

6.5.1.6 ENUM

Stereotype	ENUM
Base Class	Enumeration (from UML 1.4.2 / 2.0 meta-model)
Parent	N/A
Description	The ENUM (Enumeration Type) represents an enumeration type. The ENUM stereotype is used to define a restriction on either a content- or supplementary component of a QDT.
Tag Definitions	

Tag Definition	**Type**	**Multiplicity**	**Description**
N/A			

6.5.1.7 CodelistEntry

Stereotype	CodelistEntry
Base Class	EnumerationLiteral (from UML 1.4.2 / 2.0 meta-model)
Parent	N/A
Description	The CodelistEntry is used to stereotype entries of an enumeration with stereotype ENUM.
Tag Definitions	

Tag Definition	**Type**	**Multiplicity**	**Description**
CodeName	String	0:1	The CodeName carries the name of a certain code (like “France” in a country code list).

6.5.2 Constraints (normative)

A CDT or QDT must contain exactly one content component (an attribute of stereotype <<CON>>)

UML 1.4
```
context Classifier
      inv CDT_OneCON:
            self.isCDT or self.isQDT() implies
            self.typedFeature->select(attribute | attribute.isCON())->
            size() = 1
```

UML 2.0
```
context Class
      inv CDT_OneCON:
            self.isCDT() or self.isQDT() implies self.ownedAttribute-
            &gt;select(property |
            property.isCON())-&gt;size() = 1
```

A CDT or QDT must not contain any attributes other than those of stereotype <<CON>> or <<SUP>>.

UML 1.4
```
context Classifier
      inv CDT_onlyCONorSUP:
            self.isCDT or self.isQDT() implies
            self.typedFeature->select(attribute | not attribute.isCON() or
            attribute.isSUP())->size()=0
```

UML 2.0

```
context Class
      inv CDT_onlyCONorSUP:
            self.isCDT() or self.isQDT() implies self.ownedAttribute-
            &gt;select(property |
            not (property.isCON() or property.isSUP()))-&gt;size()=0
```

A QDT must have a dependency (with stereotype <<basedOn>>) with one CDT with target CDT.

UML 1.4

```
context Class
      inv QDT_CDTdependency:
            self.isQDT() implies
            self.supplierDependency->one(isBasedOnToCDT())
```

UML 2.0

```
context NamedElement
      inv QDT_CDTdependency:
            self.isQDT() implies self.clientDependency-&gt;exists(isBasedOnToCDT())
```

The type of CDT attributes must be a class of stereotype <<PRIM>>.

UML 1.4

```
context Classifier
      inv CDT_onlyPRIM:
            self.isCDT() implies
            self.typedFeature->forAll(attribute | attribute.isPRIM())
```

UML 2.0

```
context Class
      inv CDT_onlyPRIM:
            self.isCDT() implies self.ownedAttribute-&gt;forAll(property | prop
            erty.type.isPRIM())
```

The type of QDT attributes must be a class of stereotype either <<PRIM>> or <<ENUM>>

UML 1.4

```
context Classifier
      inv QDT_onlyPRIMorENUM:
            self.isQDT() implies
            self.typedFeature->forAll(attribute | attribute.isPRIM() or
            attribute.isENUM())
```

UML 2.0

```
context Class
      inv QDT_onlyPRIMorENUM:
            self.isQDT() implies self.ownedAttribute-&gt;forAll(property |
            property.type.isPRIM() or property.type.isENUM())
```

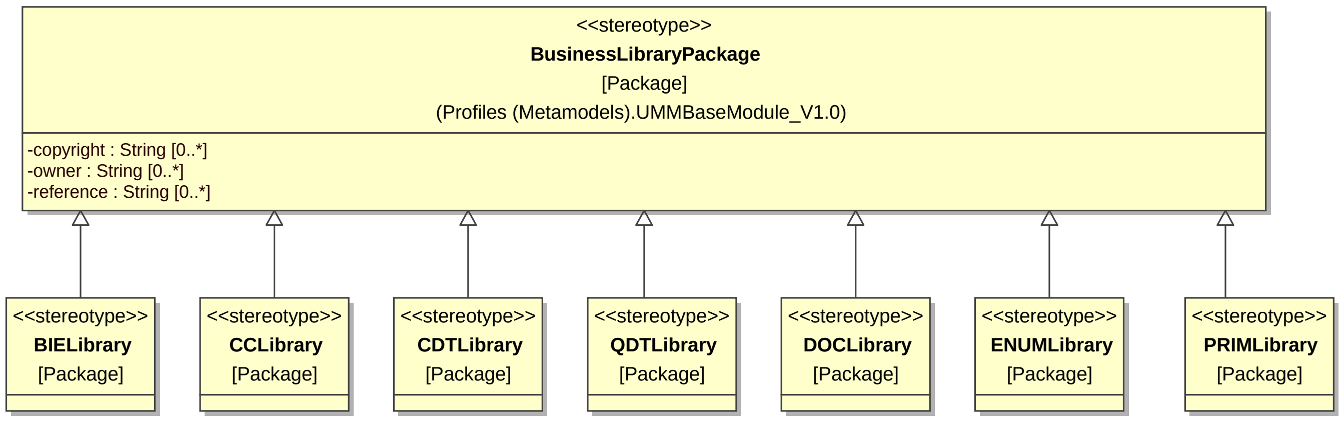
6.5.3 Data Type Stereotypes - Mapping to CCTS

CCTS Construct	**UML Implementation**
	Data Types
Core Component Type – Representation Term	Name of a UML class with stereotype <<CDT>>.
Data Type – Representation Term	Name of a UML class with stereotype <<QDT>> (without qualifier prefix if it exists). Note: The QDT class name is the qualified representation term - shall be equal to the class name of the CDT upon which this QDT is based with an optional prefix equal to the qualifier term.
Data Type – Qualifier Term	The qualifier prefix (characters before the underscore) of the QDT name.
Basis relatiosnhip between Date Type and Core Component Type	UML association of stereotype <<basedOn>>.
Content Component	UML attribute of stereotype <<CON>> of class <<CDT>> or <<QDT>>
Supplementary Component	UML attribute of stereotype <<SUP>> of class <<CDT>> or <<QDT>>
CCTS Primitive type	UML class of stereotype <<PRIM>>.
CCTS restriction	UML OCL constraint on class of stereotype <<QDT>>. CCTS restrictions act on either content or supplementary components of a QDT and the purpose is to restrict the corresponding CON/SUP of the related CDT. In this profile we use UML contraints at class level (improves visibility on the diagram) expressed using OCL. The OCL constraint itself identifies the CON/SUP on which it acts. A set of “template” OCL constraints that correspond to the allowed CCTS restriction types is provided in this specification. UML constraint “name” is taken from the CCTS controlled list and the UML constraint “body” is the OCL expression.
CCTS restriction of type “enumeration”	UML class of stereotype <<ENUM>>. Note that, so far as CCTS is concerned, “enumeration” is just one of the allowed primitives. In this profile, the enumerated primitive type is represented as a class of stereotype <<ENUM>> for convenience and readability purposes.

6.6 Management Stereotypes

6.6.1 Stereotypes and Tag Definitions (normative)

The following diagram shows a visualization of the Management Stereotypes to provide an overview of the defined elements. Note that the diagram shows only the structure of the UML Profile. Additional constraints to define compliance with the semantics of the CCTS meta-model are provided in a subsequent section.



<<stereotype>>
BusinessLibraryPackage
[Package]
(Profiles (Metamodels).UMMBaseModule_V1.0)
-copyright : String [0..*]
-owner : String [0..*]
-reference : String [0..*]
<<stereotype>> BIELibrary [Package]
<<stereotype>> CCLibrary [Package]
<<stereotype>> CDTLibrary [Package]
<<stereotype>> QDTLibrary [Package]
<<stereotype>> DOCLibrary [Package]
<<stereotype>> ENUMLibrary [Package]
<<stereotype>> PRIMLibrary [Package]

The sections below provide details on each stereotype of the Management Stereotypes. Both abstract and concrete stereotypes are described. However OCL constraints are only defined for concrete elements because these are the only ones that will exist in an actual instance model.

6.6.1.1 CC Library

Stereotype	CCLibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The CCLibrary is a container for Core Components.			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	N/A			

6.6.1.2 BIELibrary

Stereotype	BIELibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The BIELibrary is a container for Business Information Entities (BIEs).			
Tag Definitions	**Tag Definition**	**Type**	**Multipli city**	**Description**
	N/A			

6.6.1.3 CDTLibrary

Stereotype	CDTLibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The CDTLibrary is a container for the Core Data Types (CDT).			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	N/A			

6.6.1.4 QDTLibrary

Stereotype	QDTLibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The QDTLibrary is a container for Qualified Data Types (QDT).			
Tag Definitions	**Tag Definition**	**Type**	**Multi-plicity**	**Description**
	N/A			

6.6.1.5 DOCLibrary

Stereotype	DOCLibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The DOCLibrary is a container for the assembled business documents stereotyped with “RSM” which are constructed from reusable ABIEs.			
Tag Definitions	**Tag Definition**	**Type**	**Multiplicity**	**Description**
	N/A			

6.6.1.6 PRIMLibrary

Stereotype	PRIMLibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The PRIMLibrary is a container for the UN/CEFACT primitive types.			
Tag Definitions	**Tag Definition**	**Type**	**Multiplicity**	**Description**
	N/A			

6.6.1.7 ENUMLibrary

Stereotype	ENUMLibrary			
Base Class	Package (from UML 1.4.2 / 2.0 meta-model)			
Parent	Management::BusinessLibraryPackage			
Description	The ENUMLibrary is a container for enumerated types which represent code lists.			
Tag Definitions	**Tag Definition**	**Type**	**Multiplicity**	**Description**
	N/A			

6.6.2 Constraints (normative)

A CDTLibrary shall only contain classes of stereotype <<CDT>>.

UML 1.4
```
context Type
      inv CDTLibrary_onlyCDTs:
            self.isCDTLibrary() implies
            self.contents->forAll(isCDT())
```

UML 2.0
```
context Type
      inv CDTLibrary_onlyCDTs:
            self.owner.isCDTLibrary() implies self.isCDT()
```

.

A QDTLibrary shall only contain classes of stereotype <<QDT>>.

UML 1.4
```
context Type
      inv QDTLibrary_onlyQDTs:
            self.isQDTLibrary() implies
            self.contents->forAll(isQDT())
```

UML 2.0
```
context Package
      inv QDTLibrary_onlyQDTs:
            self.owner.isQDTLibrary() implies self.isQDT()
```

A CCLibrary shall only contain classes of stereotype <<ACC>> and associations of stereotype <<ASCC>>.

UML 1.4
```
context Type
      inv CCLibrary_onlyCCs:
            self.isCCLibrary() implies
            self.contents->select(content | content.isACC() or
            content.isASCC())->size()=
            self.contents->size()
```

UML 2.0
```
context Type
      inv CCLibrary_onlyCCs:
            self.owner.isCCLibrary() implies (self.isACC() or self.isASCC())
```

A BIELibrary shall only contain classes of stereotype <<ABIE>> and associations of stereotype <<ASBIE>>.

UML 1.4
```
context Type
      inv BIELibrary_onlyBIEs:
            self.isBIELibrary() implies
            self.contents->select(content | content.isABIE() or
            content.isASBIE())->size()=
            self.contents->size()
```

UML 2.0
```
context Type
      inv BIELibrary_onlyBIEs:
            self.owner.isBIELibrary() implies (self.isABIE() or self.isASBIE())
```

A ENUMLibrary shall only contain classes of stereotype <<ENUM>>.

UML 1.4
```
context Type
      inv ENUMLibrary_onlyENUMs:
            self.contents->forAll(isENUM())
```

UML 2.0
```
context Type
      inv ENUMLibrary_onlyENUMs:
            self.owner.isENUMLibrary() implies self.isENUM()
```

A DOCLibrary shall only contain classes of stereotype <<RSM>> or <<ABIE>> and accociations of stereotype <<ASBIE>>.

UML 1.4
```
context Type
      inv DOCLibrary_onlyDOCs:
            self.isDOCLibrary() implies
            self.contents->select(content | content.isRSM() or content.isABIE() or
            content.isASBIE())->size()=
            self.contents->size()
```

UML 2.0
```
context Type
      inv DOCLibrary_onlyDOCs:
            self.owner.isDOCLibrary() implies (self.isRSM() or self.isABIE() or
            self.isASBIE())
```

6.6.3 Packages & Namespace Conventions for libraries

The UPCC package structure has been designed such that one model package maps to one XML namespace in deployment. The Library stereotype inherits the "baseURN" tagged value from the registry object stereotype. Since the "baseURN" is mandatory according to the UMM Base Module it shall be present and accurate for all Library packages.

6.6.3.1 BIE Library Conventions

The name of a BIE library package can be any meaningful string but is recommended to be equal to the local name part of the baseURN. So for example, an ABIE library package might have baseURN="http://www.eurofer.org/EuroferXML/Ordering" . The name of the package would be "EuroferXML/Ordering" and it would be located in a higher level package with name (and baseURN) equal to "http://www.eurofer.org/" . In this way, the URN of any package within a nested library can be read by concatenating package names.

For CEFACT BIE libraries, the BaseURN naming MUST follow the namespace rules defined by the ATG2 XML Naming and Design Rules Specification:

<cefact namespace>:<type>:<status>:<name>:<version>   For example:

urn:un:unece:uncefact:data:standard:TBG1ReusableABIELibrary:1p.0p.1

6.6.3.2 CC Library Conventions

The core component library baseURN is defined by CEFACT and will always be the same (but with different version numbers). For example:
urn:un:unece:uncefact:data:standard:CoreComponentLibrary:0p.8p.2

6.7 OCL methods used in the UPCC Profile

1. Global OCL definitions used in the UPCC standard

UML 1.4
```
package Foundation::Core
    context ModelElement

      --Returns true if the element is a BCC
      def:
          let isBCC : Boolean =
            self.hasStereotype('BCC') and
            self.oclIsKindOf(Attribute)

      --Returns true if the element is a BBIE
      def:
          let isBBIE : Boolean =
            self.hasStereotype('BBIE') and
            self.oclIsKindOf(Attribute)
```

```
--Returns true if the element is a CON
def:
    let isCON : Boolean =
      self.hasStereotype('CON') and
      self.oclIsKindOf(Attribute)

--Returns true if the element is a SUP
def:
    let isSUP : Boolean =
      self.hasStereotype('CON') and
      self.oclIsKindOf(Attribute)

--Returns true if the element is a ACC
def:
    let isACC : Boolean =
      self.hasStereotype('ACC') and
      self.oclIsKindOf(Class)

--Returns true if the element is a ABIE
def:
    let isABIE : Boolean =
      self.hasStereotype('ABIE') and
      self.oclIsKindOf(Class)

      --Returns true if the element is an ASCC
      def:
    let isASCC : Boolean =
            self.hasStereotype('ASCC') and
            self.oclIsKindOf(Association)

      --Returns true if the element is an ASBIE
      def:
          let isASBIE : Boolean =
            self.hasStereotype('ASBIE') and
            self.oclIsKindOf(Association)

--Returns true if the element is a CDT
def:
    let isCDT : Boolean =
      self.oclIsKindOf(Class) and
      self.hasStereotype('CDT')

--Returns true if the element is a QDT
def:
    let isQDT : Boolean =
      self.oclIsKindOf(Class) and
      self.hasStereotype('QDT')

--Returns true if the element is a PRIM
def:
    let isPRIM : Boolean =
      self.oclIsKindOf(Class) and
      self.hasStereotype('PRIM')

--Returns true if the element is an ENUM
def:
    let isENUM : Boolean =
      self.oclIsKindOf(Class) and
      self.hasStereotype('ENUM')

--Returns true if the package is a CDTLibrary
def:
    let isCDTLibrary : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('CDTLibrary')

--Returns true if the package is a QDTLibrary
def:
    let isQDTLibrary : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('QDTLibrary')

--Returns true if the package is a CCLibrary
def:
    let isCCLibrary : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('CCLibrary')

--Returns true if the package is a BIELibrary
def:
    let isBIELibrary : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('BIELibrary')

--Returns true if the package is a ENUMLibrary
def:
    let isENUMLibrary : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('ENUMLibrary')

--Returns true if the package is a DOCLibrary
def:
    let isDOCLibrary : Boolean =
      self.oclIsKindOf(Package) and
      self.hasStereotype('DOCLibrary')

--Returns true if the element has a stereotype equal to the
      --passed string
def:
    let hasStereotype(st : String) : Boolean =
            self.stereotype->select(self.name=st)->notEmpty()

      --checks if a dependency is stereotyped as basedOn and leads to
      --an ACC
      def:
    let isBasedOnToACC : Boolean =
            self.oclIsKindOf(Dependency) and
            self.hasStereotype('basedOn') and
            self.oclAsType(Dependency).client->
            select(client | client.isACC())->size()=1

      -- checks if a dependency is stereotyped as basedOn and leads
      --to a CDT
      def:
    let isBasedOnToCDT : Boolean =
            self.oclIsKindOf(Dependency) and
            self.hasStereotype('basedOn') and
            self.oclAsType(Dependency).client->
            select(client | client.isCDT())->size()=1

      -- returns true if the association type is "Composition"
      def:
    let isComposition() : Boolean =
            self.oclIsKindOf(AssociationEnd) and
```

```
                  self.oclAsType(AssociationEnd).aggregation =
                  AggregationKind::composite

endpackage
```

UML 2.0

```
package Classes::Kernel
    context Element

      --Returns true if the element is a BCC
      def:
            let isBCC : Boolean =
                  self.oclAsType(Property).extension_BCC.isDefined

      --Returns true if the element is a BBIE
      def:
            let isBBIE : Boolean =
                  self.oclAsType(Property).extension_BBIE.isDefined

      --Returns true if the element is a CON
      def:
            let isCON : Boolean =
                  self.oclAsType(Property).extension_CON.isDefined

      --Returns true if the element is a SUP
      def:
            let isSUP : Boolean =
                  self.oclAsType(Property).extension_SUP.isDefined

      --Returns true if the element is a ACC
      def:
            let isACC : Boolean =
                  self.oclAsType(Class).extension_ACC.isDefined

      --Returns true if the element is a ABIE
      def:
            let isABIE : Boolean =
                  self.oclAsType(Class).extension_BIE.
                  self.oclAsType(Class).extension_ABIE.isDefined

      --Returns true if the element is an ASCC
      def:
            let isASCC : Boolean =
                  self.oclAsType(Association).extension_ASCC.isDefined

      --Returns true if the element is an ASBIE
      def:
            let isASBIE : Boolean =
                  self.oclAsType(Class).extension_BIE.
                  self.oclAsType(Association).extension_ASBIE.isDefined

      --Returns true if the element is a SUP
      def:
            let isRSM : Boolean =
                  self.oclAsType(Class).extension_RSM.isDefined

      --Returns true if the element is a CDT
      def:
            let isCDT : Boolean =
                  self.oclAsType(Class).extension_CDT.isDefined
```

```
--Returns true if the element is a QDT
def:
      let isQDT : Boolean =
            self.oclAsType(Class).extension_BIE.
            self.oclAsType(Class).extension_QDT.isDefined

--Returns true if the element is a PRIM
def:
      let isPRIM : Boolean =
            self.oclAsType(Class).extension_PRIM.isDefined


--Returns true if the element is an ENUM
def:
      let isENUM : Boolean =
            self.oclAsType(Class).extension_ENUM.isDefined

--Returns true if the package is a CDTLibrary
def:
      let isCDTLibrary : Boolean =
            self.oclAsType(Package).extension_RegistryObject.
            oclIsKindOf(UMLprofileforCoreComponents20070928__CDTLibrary)

--Returns true if the package is a QDTLibrary
def:
      let isQDTLibrary : Boolean =
            self.oclAsType(Package).extension_RegistryObject
            .oclIsKindOf(UMLprofileforCoreComponents20070928__QDTLibrary)


--Returns true if the package is a CCLibrary
def:
      let isCCLibrary : Boolean =
            self.oclAsType(Package).extension_RegistryObject
            .oclIsKindOf(UMLprofileforCoreComponents20070928__CCLibrary)

--Returns true if the package is a BIELibrary
def:
      let isBIELibrary : Boolean =
            self.oclAsType(Package).extension_RegistryObject
            .oclIsKindOf(UMLprofileforCoreComponents20070928__BIELibrary)

--Returns true if the package is a ENUMLibrary
def:
      let isENUMLibrary : Boolean =
            self.oclAsType(Package).extension_RegistryObject
            .oclIsKindOf(UMLprofileforCoreComponents20070928__ENUMLibrary)

--Returns true if the package is a DOCLibrary
def:
      let isDOCLibrary : Boolean =
            self.oclAsType(Package).extension_RegistryObject
            .oclIsKindOf(UMLprofileforCoreComponents20070928__DOCLibrary)


-- checks if a dependency is stereotyped as basedOn and leads to an ACC
def:
      let isBasedOnToACC : Boolean =
            self.oclIsKindOf(Dependency) and
            self.oclAsType(Dependency).extension_basedOn.isDefined and
            self.oclAsType(Dependency).supplier-&gt;select(supplier | sup
            plier.isACC())-&gt;size()=1
      -- checks if a dependency is stereotyped as basedOn and leads to a CDT
      def:
            let isBasedOnToCDT : Boolean =
                  self.oclIsKindOf(Dependency) and
                  self.oclAsType(Dependency).extension_basedOn.isDefined and
                  self.oclAsType(Dependency).supplier-&gt;select(supplier | sup
                  plier.isCDT())-&gt;size()=1

      -- checks if a dependency is stereotyped as basedOn
      def:
            let isBasedOn : Boolean =
                  self.oclIsKindOf(Dependency) and
                  self.oclAsType(Dependency).extension_basedOn.isDefined

      -- checks if a dependency is stereotyped as basedOn
      def:
            let realAttributes : Sequence(Property =
                  self.ownedAttribute-&gt;select(attr|attr.association.isUndefined)

endpackage
```

6.8 Context Package – UML Profile

UML based concepts to represent context will be added based on the results of the Unified Context Methodology (UCM) project from TMG.

6.9 Relationship to UMM

The UMM Business Transaction View package contains sub packages of type <<BusinessInformationView>>. This package should contain only a class of stereotype <<InformationEnvelope>> element that has an association to an RSM in a <<DOCLibrary>> package[1]. The <<DocLibrary>> package, in turn contains one or more assemply documents (RSM) that are constructed from ABIEs in packages of stereotype <<BIELibrary>>

This approach is designed to separate three key namespaces:

- The namespace of the UMM Business process (typically realized in a BPSS, WS-CDL or abstract BPEL schema).
- The namespace of the Business Information that is used in the business process (typically realised as an XSD schema)
- The namespace of the ABIE library components that are used to build the business information (typically realised as re-usable XSD schema that are imported into the root schema)

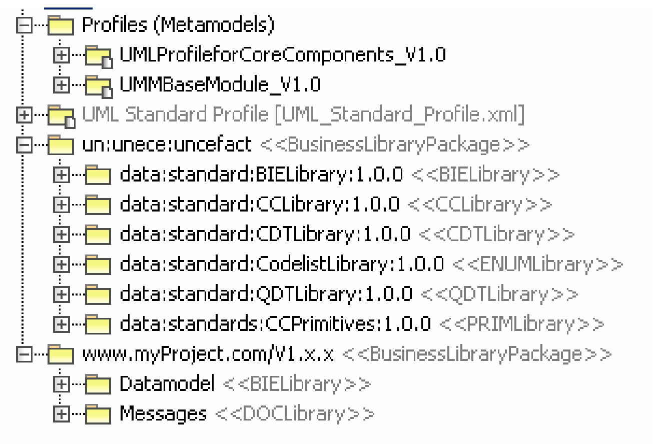
[1] A final solution to message assembly will be provided by the Core Components Message Assembly Project (CCMA) project.

7 Example

This section is *informative*. It is provided as a guide to the business modeller on best practices for CCTS modelling using this UML profile, illustrated with examples.

7.1 Structure

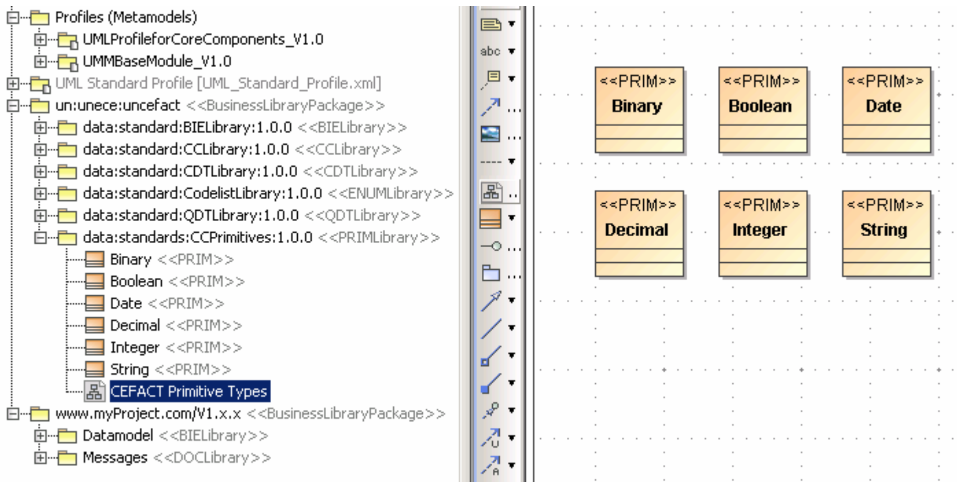
An information model based on the UML profile for CCTS can contain UML packages with the stereotypes “DocLibrary”, “BIELibrary”, “CCLibrary”, QDTLibrary”, “CDTLibrary”, “ENUMLibrary” and “PRIMLibrary”. These packages can be further structured within packages with stereotype “BusinessLibraryPackage”. The following figure shows the packages of an example model based on the UML profile for CCTS. This example will be explained in the subsequent sections.



```
Profiles (Metamodels)
    UMLProfileforCoreComponents_V1.0
    UMMBaseModule_V1.0
UML Standard Profile [UML_Standard_Profile.xml]
un:unece:uncefact <<BusinessLibraryPackage>>
    data:standard:BIELibrary:1.0.0 <<BIELibrary>>
    data:standard:CCLibrary:1.0.0 <<CCLibrary>>
    data:standard:CDTLibrary:1.0.0 <<CDTLibrary>>
    data:standard:CodelistLibrary:1.0.0 <<ENUMLibrary>>
    data:standard:QDTLibrary:1.0.0 <<QDTLibrary>>
    data:standards:CCPrimitives:1.0.0 <<PRIMLibrary>>
www.myProject.com/V1.x.x <<BusinessLibraryPackage>>
    Datamodel <<BIELibrary>>
    Messages <<DOCLibrary>>
```

7.2 PRIMLibrary

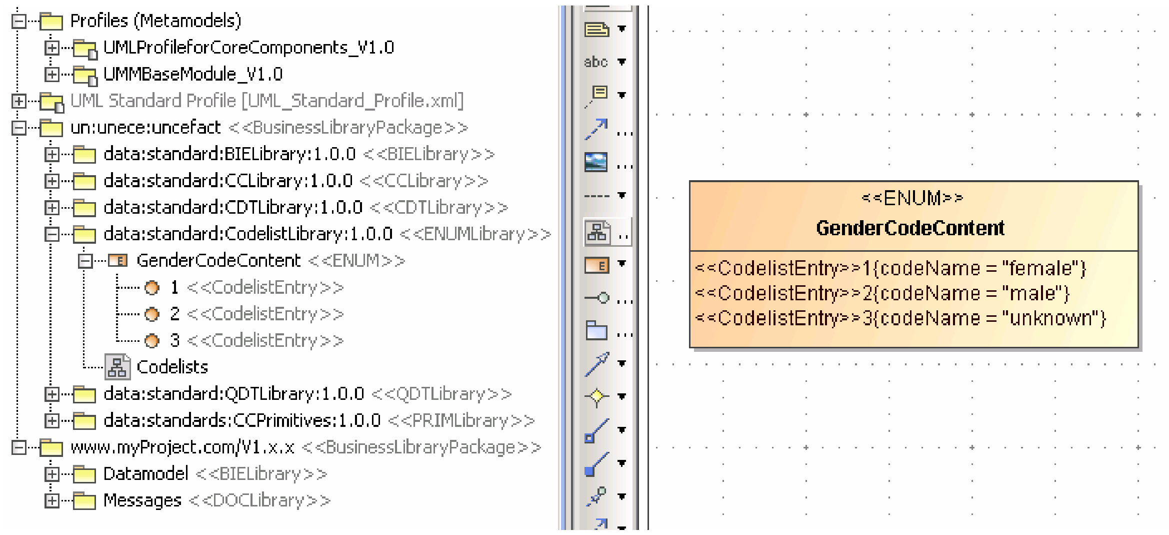
The package with the stereotype “PRIMLibrary” contains the fixed set of CEFACT primitive types as defined in the CCTS. The CEFACT primitive types are later on used to provide primitive types for the content components (CON) and supplementary components (SUP) of the core data types (CDT) and qualified data types (QDT) and are represented as UML classes with the stereotype “PRIM”:



```
Profiles (Metamodels)
    UMLProfileforCoreComponents_V1.0
    UMMBaseModule_V1.0
UML Standard Profile [UML_Standard_Profile.xml]
un:unece:uncefact <<BusinessLibraryPackage>>
    data:standard:BIELibrary:1.0.0 <<BIELibrary>>
    data:standard:CCLibrary:1.0.0 <<CCLibrary>>
    data:standard:CDTLibrary:1.0.0 <<CDTLibrary>>
    data:standard:CodelistLibrary:1.0.0 <<ENUMLibrary>>
    data:standard:QDTLibrary:1.0.0 <<QDTLibrary>>
    data:standards:CCPrimitives:1.0.0 <<PRIMLibrary>>
        Binary <<PRIM>>
        Boolean <<PRIM>>
        Date <<PRIM>>
        Decimal <<PRIM>>
        Integer <<PRIM>>
        String <<PRIM>>
        CEFACT Primitive Types
www.myProject.com/V1.x.x <<BusinessLibraryPackage>>
    Datamodel <<BIELibrary>>
    Messages <<DOCLibrary>>
```

7.3 ENUMLibrary

The package with the stereotype “ENUMLibrary” contains the enumerations (code lists) that are used within the model. Enumerations may later on used to restrict the content component (CON) of qualified data types (QDT) and are represented as UML enumerations with the stereotype “ENUM”. Note that in most cases enumerations will be maintained in XML-Schemas and not in the UML model (in UML only for rather short lists like a gender code or restricted country codelists).

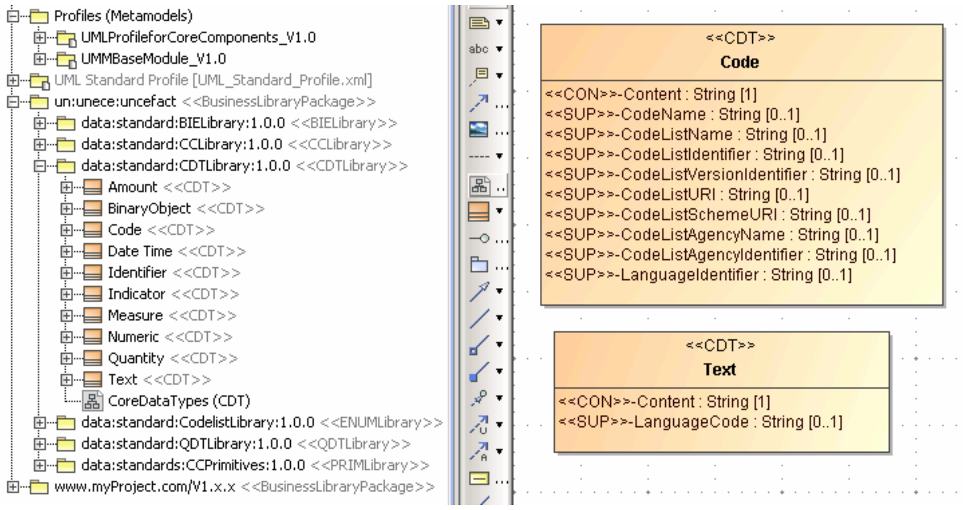


```
Profiles (Metamodels)
  UMLProfileforCoreComponents_V1.0
  UMMBaseModule_V1.0
UML Standard Profile [UML_Standard_Profile.xml]
un:unece:uncefact <<BusinessLibraryPackage>>
  data:standard:BIELibrary:1.0.0 <<BIELibrary>>
  data:standard:CCLibrary:1.0.0 <<CCLibrary>>
  data:standard:CDTLibrary:1.0.0 <<CDTLibrary>>
  data:standard:CodelistLibrary:1.0.0 <<ENUMLibrary>>
    GenderCodeContent <<ENUM>>
      1 <<CodelistEntry>>
      2 <<CodelistEntry>>
      3 <<CodelistEntry>>
    Codelists
  data:standard:QDTLibrary:1.0.0 <<QDTLibrary>>
  data:standards:CCPrimitives:1.0.0 <<PRIMLibrary>>
www.myProject.com/V1.x.x <<BusinessLibraryPackage>>
  Datamodel <<BIELibrary>>
  Messages <<DOCLibrary>>
```

```
<<ENUM>>
GenderCodeContent
<<CodelistEntry>>1{codeName = "female"}
<<CodelistEntry>>2{codeName = "male"}
<<CodelistEntry>>3{codeName = "unknown"}
```

7.4 CDTLibrary

The package with the stereotype “CDTLibrary” contains the fixed set of core component types as defined in the CCTS. The core component types are later on used to derive qualified data types (QDT) by restriction and are represented as UML classes with the stereotype “CDT” containing exactly one content component as a UML attribute with the stereotype “CON” and one to many supplementary components as UML attributes with the stereotype “SUP”.



```
Profiles (Metamodels)
  UMLProfileforCoreComponents_V1.0
  UMMBaseModule_V1.0
UML Standard Profile [UML_Standard_Profile.xml]
un:unece:uncefact <<BusinessLibraryPackage>>
  data:standard:BIELibrary:1.0.0 <<BIELibrary>>
  data:standard:CCLibrary:1.0.0 <<CCLibrary>>
  data:standard:CDTLibrary:1.0.0 <<CDTLibrary>>
    Amount <<CDT>>
    BinaryObject <<CDT>>
    Code <<CDT>>
    Date Time <<CDT>>
    Identifier <<CDT>>
    Indicator <<CDT>>
    Measure <<CDT>>
    Numeric <<CDT>>
    Quantity <<CDT>>
    Text <<CDT>>
    CoreDataTypes (CDT)
  data:standard:CodelistLibrary:1.0.0 <<ENUMLibrary>>
  data:standard:QDTLibrary:1.0.0 <<QDTLibrary>>
  data:standards:CCPrimitives:1.0.0 <<PRIMLibrary>>
www.myProject.com/V1.x.x <<BusinessLibraryPackage>>
```

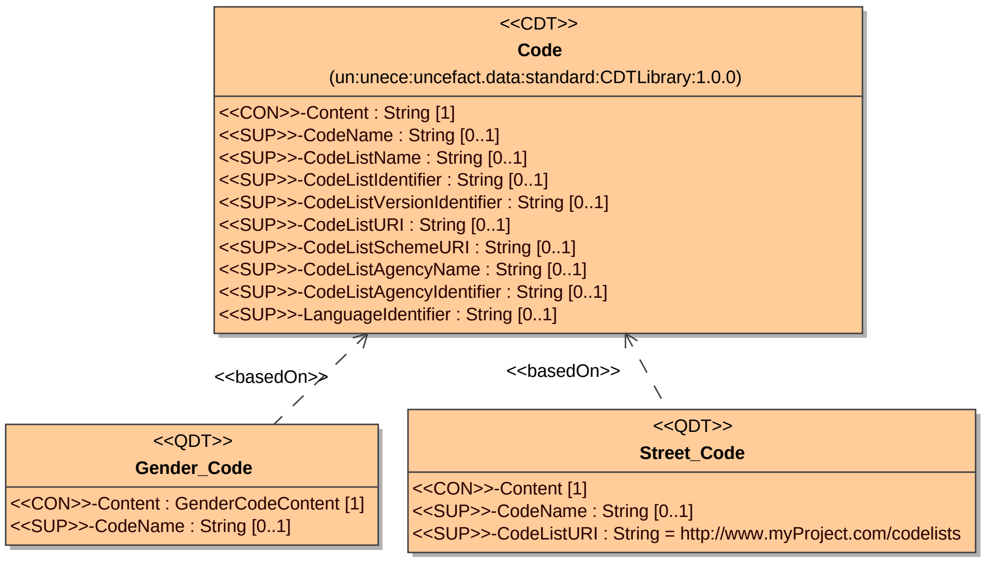
```
<<CDT>>
Code
<<CON>>-Content : String [1]
<<SUP>>-CodeName : String [0..1]
<<SUP>>-CodeListName : String [0..1]
<<SUP>>-CodeListIdentifier : String [0..1]
<<SUP>>-CodeListVersionIdentifier : String [0..1]
<<SUP>>-CodeListURI : String [0..1]
<<SUP>>-CodeListSchemeURI : String [0..1]
<<SUP>>-CodeListAgencyName : String [0..1]
<<SUP>>-CodeListAgencyIdentifier : String [0..1]
<<SUP>>-LanguageIdentifier : String [0..1]
```

```
<<CDT>>
Text
<<CON>>-Content : String [1]
<<SUP>>-LanguageCode : String [0..1]
```

7.5 QDTLibrary

The package with the stereotype “QDTLibrary” contains the qualified data types (QDT) used within the model. The qualified data types are derived by restriction from CDTs and are represented as UML classes with the stereotype “QDT” containing exactly one content component as a UML attribute with the stereotype “CON” and zero to many supplementary components as UML attributes with the stereotype “SUP”.

The fact that a QDT is derived from a CDT is represented as a UML dependency with stereotype “basedOn” between a QDT and a CDT. The name of a QDT contains the name of the CDT it is based on prefixed by zero to many semantic qualifiers separated with underscore as defined in the CCTS.



<<CDT>>
Code
(un:unece:uncefact.data:standard:CDTLibrary:1.0.0)
<<CON>>-Content : String [1]
<<SUP>>-CodeName : String [0..1]
<<SUP>>-CodeListName : String [0..1]
<<SUP>>-CodeListIdentifier : String [0..1]
<<SUP>>-CodeListVersionIdentifier : String [0..1]
<<SUP>>-CodeListURI : String [0..1]
<<SUP>>-CodeListSchemeURI : String [0..1]
<<SUP>>-CodeListAgencyName : String [0..1]
<<SUP>>-CodeListAgencyIdentifier : String [0..1]
<<SUP>>-LanguageIdentifier : String [0..1]
<<basedOn>>
<<basedOn>>
<<QDT>>
Gender_Code
<<CON>>-Content : GenderCodeContent [1]
<<SUP>>-CodeName : String [0..1]
<<QDT>>
Street_Code
<<CON>>-Content [1]
<<SUP>>-CodeName : String [0..1]
<<SUP>>-CodeListURI : String = http://www.myProject.com/codelists

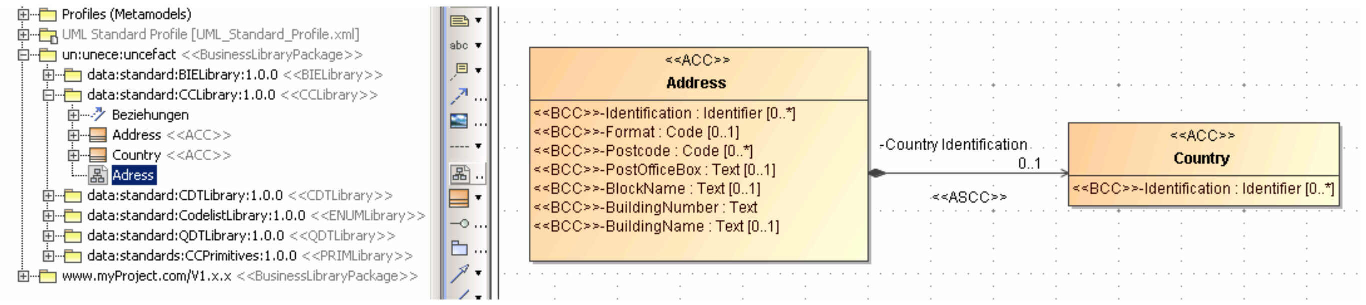
A qualified datatype based on CDT “Code” can for example be restricted by assigning a specific (runtime-) codeliste to the content component (like GenderCodeContent in Gender_Code) or by providing a URL in the supplementary component “CodeListURI” which points to a place where the respective codelist can be found.

Note that the qualified data types in the example are restricted by means of:
- restriction of the content component (CON) by assigning an enumeration from an “ENUMLibrary”,
- restriction of the content component (CON) by assigning a written constraint,
- removing supplementary components (SUP) or
- assigning default values to supplementary components (SUP).

If there is no additional support from the UML tool in use, a QDT is created by
- copy (duplicate) the (to be restricted) CDT into a package with stereotype “QDTLibrary”,
- change the stereotype from CDT to QDT,
- draw a dependency with the stereotype “basedOn” between the QDT and the CDT it is based on and
- define the QDT by restricting it and optionally adding semantic qualifiers.

7.6 CCLibrary

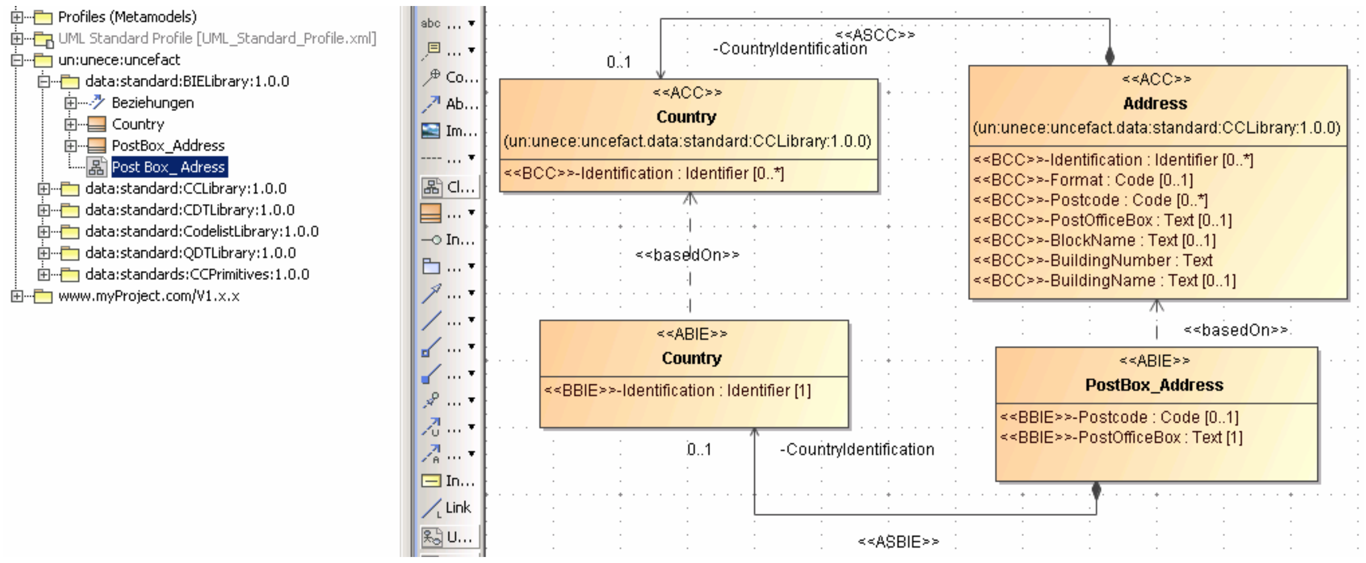
The package with the stereotype “CCLibrary” contains aggregate core components represented as UML classed with stereotype “ACC” consisting of basic core components represented as UML attributes with stereotype “BCC” and association core components represented as UML compositions with stereotype “ASCC”. Aggregate core components are generic information objects as defined in the CCTS. The aggregate core components (ACC) are later on used to derive aggregate business information entities (ABIE) by restriction. The name of a class with stereotype “ACC” is the object class term as defined in the CCTS while the names the attributes with stereotype “BCC” are the property terms and the target roles of compositions with stereotype “ASCC” are the representation terms (not the dictionary entry names respectively). The types of the attributes with stereotype “BCC” are taken from the “CDTLibrary”. The given example uses a small part of the library published by TBG17:



Profiles (Metamodels)
UML Standard Profile [UML_Standard_Profile.xml]
un:unece:uncefact <<BusinessLibraryPackage>>
data:standard:BIELibrary:1.0.0 <<BIELibrary>>
data:standard:CCLibrary:1.0.0 <<CCLibrary>>
Beziehungen
Address <<ACC>>
Country <<ACC>>
Adress
data:standard:CDTLibrary:1.0.0 <<CDTLibrary>>
data:standard:CodelistLibrary:1.0.0 <<ENUMLibrary>>
data:standard:QDTLibrary:1.0.0 <<QDTLibrary>>
data:standards:CCPrimitives:1.0.0 <<PRIMLibrary>>
www.myProject.com/V1.x.x <<BusinessLibraryPackage>>
<<ACC>>
Address
<<BCC>>-Identification : Identifier [0..*]
<<BCC>>-Format : Code [0..1]
<<BCC>>-Postcode : Code [0..*]
<<BCC>>-PostOfficeBox : Text [0..1]
<<BCC>>-BlockName : Text [0..1]
<<BCC>>-BuildingNumber : Text
<<BCC>>-BuildingName : Text [0..1]
-Country Identification
0..1
<<ASCC>>
<<ACC>>
Country
<<BCC>>-Identification : Identifier [0..*]

7.7 BIELibrary

The package with the stereotype “BIELibrary” contains aggregate business information entities represented as UML classed with stereotype “ABIE” consisting of basic business information entities represented as UML attributes with stereotype “BBIE” and association business information entities represented as UML compositions with stereotype “ASBIE”.


Profiles (Metamodels)
UML Standard Profile [UML_Standard_Profile.xml]
un:unece:uncefact
data:standard:BIELibrary:1.0.0
Beziehungen
Country
PostBox_Address
Post Box_ Adress
data:standard:CCLibrary:1.0.0
data:standard:CDTLibrary:1.0.0
data:standard:CodelistLibrary:1.0.0
data:standard:QDTLibrary:1.0.0
data:standards:CCPrimitives:1.0.0
www.myProject.com/V1.x.x
<<ASCC>>
-CountryIdentification
0..1
<<ACC>>
Country
(un:unece:uncefact.data:standard:CCLibrary:1.0.0)
<<BCC>>-Identification : Identifier [0..*]
<<ACC>>
Address
(un:unece:uncefact.data:standard:CCLibrary:1.0.0)
<<BCC>>-Identification : Identifier [0..*]
<<BCC>>-Format : Code [0..1]
<<BCC>>-Postcode : Code [0..*]
<<BCC>>-PostOfficeBox : Text [0..1]
<<BCC>>-BlockName : Text [0..1]
<<BCC>>-BuildingNumber : Text
<<BCC>>-BuildingName : Text [0..1]
<<basedOn>>
<<basedOn>>
<<ABIE>>
Country
<<BBIE>>-Identification : Identifier [1]
<<ABIE>>
PostBox_Address
<<BBIE>>-Postcode : Code [0..1]
<<BBIE>>-PostOfficeBox : Text [1]
0..1
-CountryIdentification
<<ASBIE>>

ABIEs are are reusable information objects that are derived by restriction from ACCs. The fact that an ABIE is derived from an ACC is represented as a UML dependency with stereotype “basedOn” between an ABIE and an ACC.

The names of ABIEs, BBIEs and ASBIEs contain the name of the respective ACC, BCC and ASCC they are based on prefixed by zero to many semantic qualifiers separated with underscore as defined in the CCTS.

Note that the classes with stereotype “ABIE” can be restricted by means of:

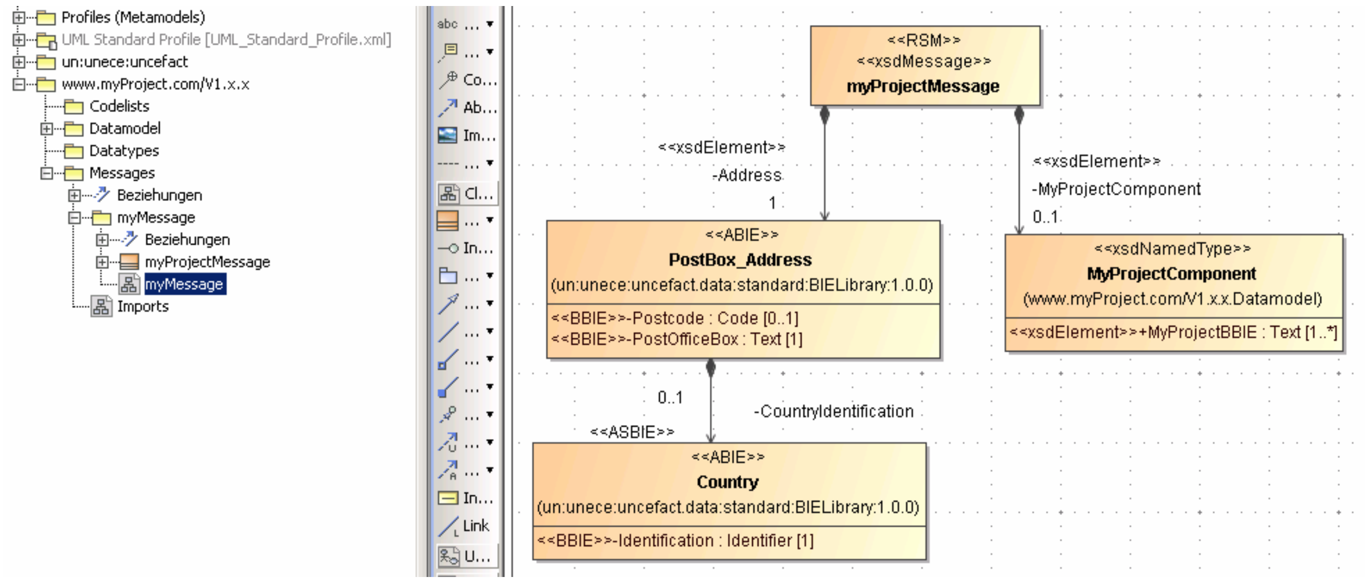
- restricted number of attributes (BBIEs) and compositions (ASBIEs),
- restricted multiplicity of attributes (BBIEs) and compositions (ASBIEs) and
- restricted data types using qualified data types (QDT) as restricted CDTs to type attributes (BBIEs).

If there is no additional support from the UML tool in use, an ABIE is created by

- copy (duplicate) the (to be restricted) ACC into a package with stereotype “ABIELibrary”,
- define the ABIE by restricting it (e.g. set multiplicity by defining a single UML Multiplicity Range for the attribute, change the data type by assigning the type of the attribute to a class stereotyped with “QDT”),
- add semantic qualifiers,
- change the remaining stereotypes from ACC to ABIE, from BCC to BBIE and from ASCC to ASBIE and
- draw a dependency with the stereotype “basedOn” between the ABIE and the ACC it is based on.

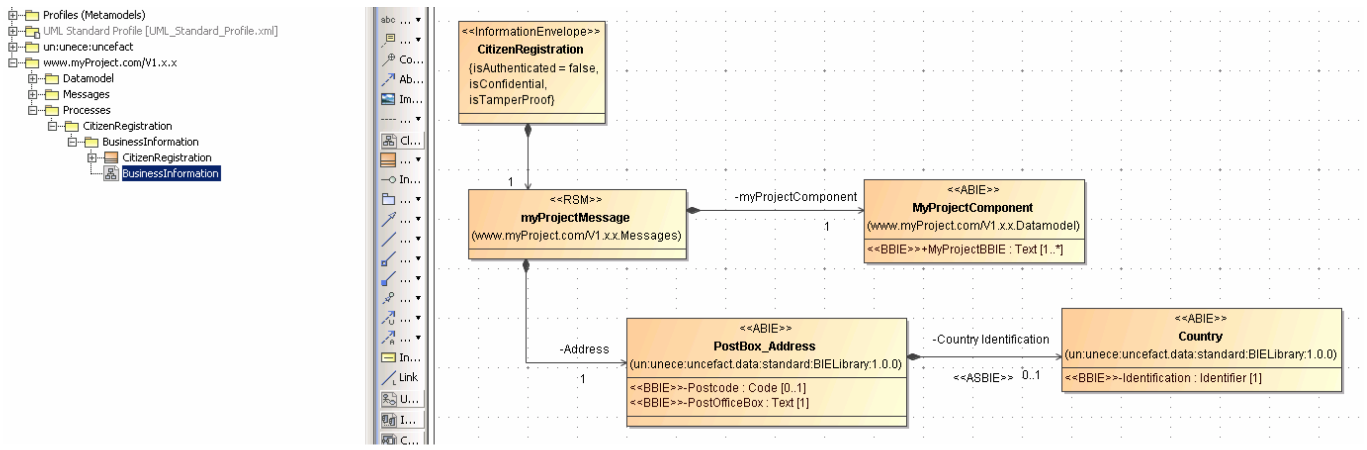
7.8 DocLibrary

The package with the stereotype “DocLibrary” contains the business information assembled from reusable ABIEs as Root Schema Modules (RSM) to be exchanged in a given business scenario.



7.9 Proposed link to UMM artefacts (UMM Foundation 1.0 – Final Specification)

Business information from a “DOCLibrary” can be used as part of an UMM collaboration model within the “BusinessInformationView” (see UMM Foundation Module 1.0 for further information):



8 Recommended XMI-interchange-format for Core-Components UML-models based on UPCC: EMF UML2 XMI-format

The XMI-specification is a procedure that defines how an instance of an arbitrary MOF meta model is to be stored as an XML representation. Thus XMI varies with the MOF version, the UML version and the version of the XMI standard used. All combinations of these factors represent legal and practiced versions of UML XMI. Two MOF versions of the 1.x variety, potentially four UML Versions of the 1.x variety and two revisions of the XMI standard can provide up to 16 different legal types of XMI.

In order to achieve interoperability there must be a point of reference to define a notion of correctness for XMI-encoded UML and thus UPCC-based Core Component models.

One manner of solving this problem is to define constraints on XMI using an XML-based mechanism such as DTD, Schema or relax-ng and publish these constraints as a "XMI-Profile". Vendors may then implement this XMI-flavour to ensure compliance. It is important to note, that such an approach defines one specific combination of UML, XMI and MOF which might already exclude certain tools from being capable of ensuring interoperability. In addition, if the underlying UML is restricted to only cover a certain area of the UML (e.g. class diagrams), as has been done in other specifications, implicit semantic constraints on the language are implicitly introduced. This is explicitly prohibited by the OMG document defining the use of profiles. To comply with an approach along the lines of such a specifications, vendors are forced to write transformations and filters to alter their XMI in this very specific manner. Consequently, it is not certain that vendors will be inclined to implement such a complex and costly transformation for the UN/CEFACT market.

Therefore the current specification refers to an existing XMI-flavour that is already recognised and accepted by the marketplace in order to define the UPCC XMI for the exchange of Core-Components models.

Since the Eclipse Modeling Framework (EMF) gains more and more traction, the XMI-Format accepted by the Eclipse UML2 reference implementation based on XMI 2.1 (http://schema.omg.org/spec/XMI/2.1) and UML 2.0 (http://www.eclipse.org/uml2/2.0.0/UML) is used as the UPCC reference. This format is referred to as “EMF UML2 XMI” in this document.

Tests in Implementation Verification have shown that UPCC-profiled UML models can be exchanged between different UML tools based on EMF UML2 XMI (including Magic Draw 12.5, IBM Rational Software Modeler 7.0, Visual Paradigm 6.0 SP2, Omondo Eclipse UML Studio Edition 2.1.0, Papyrus UML 1.7[2]). Models from Enterprise-Architect 6.x and 7 can be exported in EMF UML2 XMI-format based on an external export filter from http://www.openarchitectureware.org (not tested).

9 Conformance Criteria

This section is *normative*.

Conformance statements conform to the requirements of the W3C QA guidelines:

http://www.w3.org/TR/2004/WD-qaframe-spec-20041122/

9.1 Product Classes

The following types of product may claim comliance with this specification:

- UML Modelling tools from tool vendors may claim compliance.
- UML Model instances from working groups (eg TBG) may claim compliance.

[2] The test setup included the export of a fully profiled Core-Component model from Magic Draw in EMF UML2 XMI-format and the import into the other mentioned UML tools.

- External Validation services (eg ICG repository service) may claim compliance.

9.2 Profiles and Levels

The W3C Quality Assurance guidelines allow the specification of different level of compliance. For UPCC this can mean

- Compliance only with the structural components (stereotypes & tagged values).
- Or compliance with the comlete specification including the semantic constraints defined by the OCL constraints.

10 Reference Implementations

Two reference implementations of the UPCC specification have been developed as part of the Implementation Verification phase of this project:

- University of Vienna / Research Studios Austria: Impementation based on a plug-in for the UML-tool Enterprise Architect ("UMM Add-In"). The UMM Add-In supports validation of UML-models in Enterprise Architect (EA) format against the UPCC-constraints (implemented based on the EA-API) and generation of ATG NDR conformant XML-Schemas and can be downloaded at http://ummaddin.researchstudio.at/).

- Germany-Online Standardization project: Implementation based on the Eclipse Modeling Framework (EMF) and the Eclipse UML2-project ("XGenerator"). The XGenerator supports validation of UML models in EMF UML2 XMI-format against the UPCC-constraints (implemented as OCL in an external text file included at runtime) and generation of XML-Schema, docbook and other deployment artefacts based on Velocity templates. XGenerator is availabe at http://sourceforge.net/.

The Eclipse-based reference implementation is publicly available as open source and can consequently be used as a point of reference for validation of UPCC conformant models and also as a validation service.