Core Components Realisation

eBTWG

1st December 2001
Version 1.00

DRAFT FOR REVIEW

Part 1 – Relation to ebXML Architecture

Requirements, Roles, Deliverables

1 Introduction

Within the ebXML technical architecture there is a key element known as the ebXML Registry[1]. There is a need to allow data components and associated meta information to be loaded and extracted from this registry in a format that maintains the full set of metadata that expresses the semantics of those data components. Along side this requirement there is also a need for ebXML based processes to utilize default assembly mappings from this extract format into useful e-business interchange payload definitions. There is also a need to explain the process of how existing industry component libraries can be enhanced to become candidates for adoption into UN/CEFACT core component libraries and then potentially promoted as wider cross-industry core components in their own right.

The Core Component Realisation group (CCR) will define the Import and Export format and the default mappings into XML for exported components. This format detail and its interaction with the Registry constitute the Component Registry Interface (CRI). The team will work with groups both inside the ebTWG and also those industry and standards groups that have submitted useful contributions to the overall core component work within UN/CEFACT.

Within the ebTWG effort itself the CCR will need to work with both the UML-to-XML group and the Core Components group, and in the broader context with the continuing ebXML Registry work also.

This means that the CCR group expected to have the following deliverables:

1) The definition of the Component Registry Interface (CRI). This is contained within this document.

2) The definition of the mappings from the CRI to useful payload assembly definitions (Registry to Payload - R2P). This will be documented in a separate document. The initial thoughts related to this are to try to use UML model interchange syntax specification - XMI version 2 - as the basis for this format. This will allow migration too and from different syntax formats and also it is hoped will allow easy integration with UML modelling tools. Potentially XMI Version 2 would also enable a format that would also document the meta-data that has been capture and loaded into the repository.

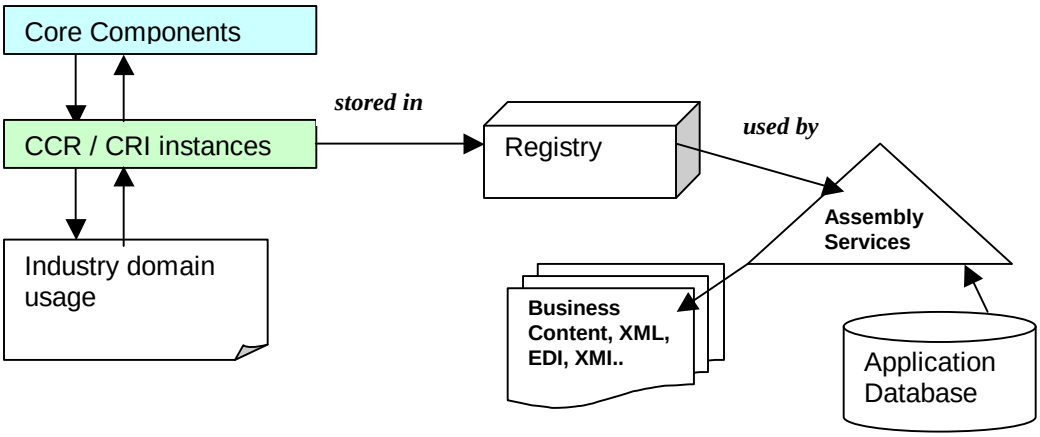
3) The process for the conversion of existing component libraries to allow them to be loaded into the Registry and how they can utilise the existing UN Components and also to be elevated to UN Component Status. Once loaded these components can then be enhanced to refer to Core Components, or established as actual UN certified

[1] Older documentation may sometimes also refer to Registry/Repository, or Reg/Rep for short.

Core Components themselves (UNCC). This is not the same as the Core Component Discovery process; it is a support process subsequent to that discovery phase.

Additionally this approach ensures that the core component semantics are therefore being captured and stored in a neutral simple XML instance structure that is not specific to anyone rendering or dialect for business documents. This is critical to ensure future adaptability and to avoid reliance on any short-term syntax devices. The document formats required for business interactions (verbs and transactions) themselves are then render as either XML, or any other convenient format such as EDI as needed by the business process implementation and trading partner requirements. These format assembly instructions are also stored associated with the assembly core component definitions and can be provided and accessed by vendors and implementers as needed.

Figure 1. Relationships of CCR / CRI components



The distinction therefore between components in the registry and between the UN sponsored Core Components (UNCC) is that the UNCC have reached a semantic quality that allows them to be used interoperably. However by allowing organizations to migrate their existing legacy Component Libraries these therefore have a significant role to play in increasing the number of components available as candidates for future adoption as UN Core Components. Figure 2 below illustrates this process.

Another consideration is to be able to provide consistent document format instructions for business process implementation. Providing a default library of document format artefacts that industries can standardize on to enable consistent and reliable document interactions and reduce the need for transformation services.

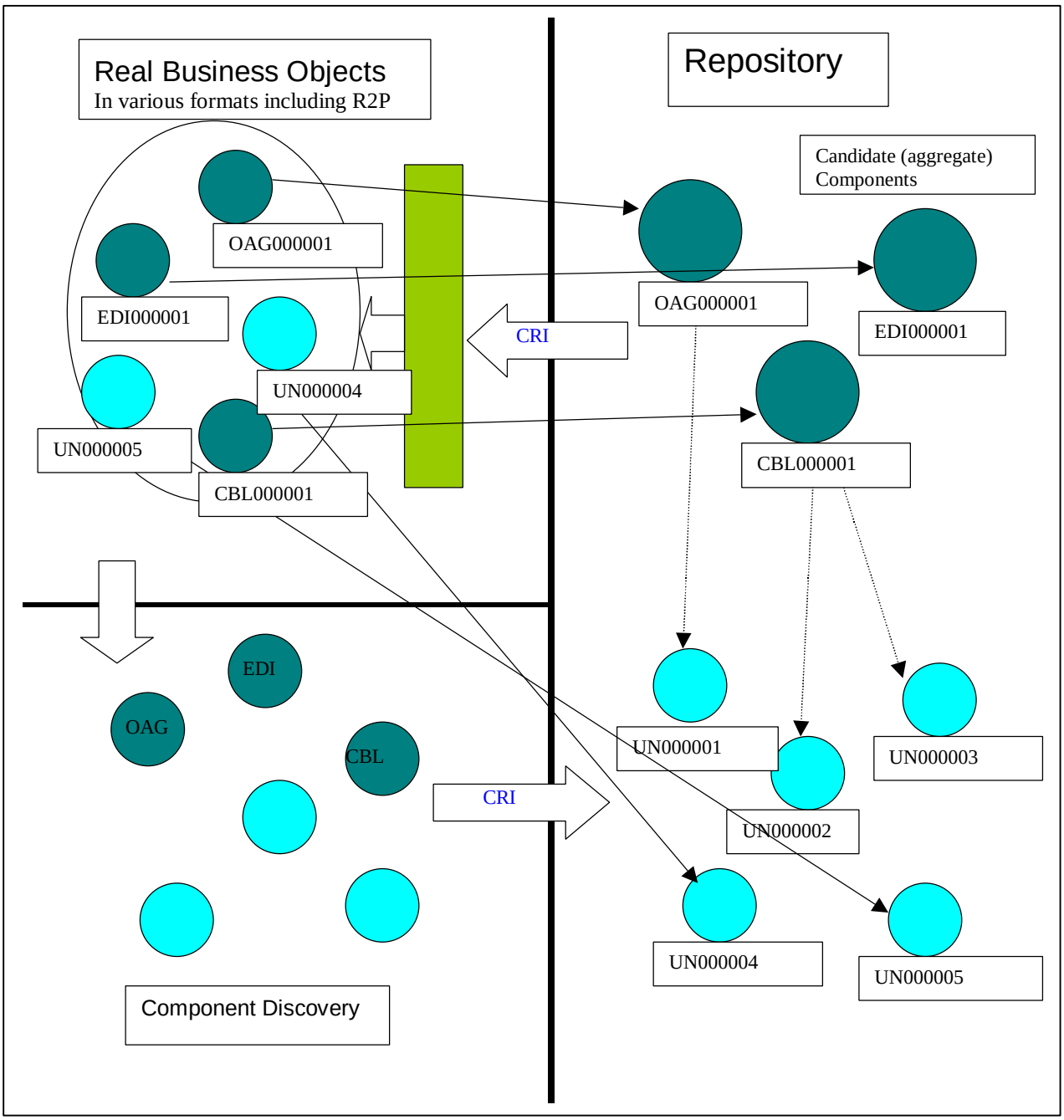
This CRI work is designed to provide the technical facilitation that the other ebTWG groups need to then work logically with to refine and deliver the business artefacts that the end user organizations ultimately need to implement ebXML-based systems. This work is key to linking the top down approach of the Business Process team and the bottom up approach of the Core Component team.

Simply put the CCR project deliverables enables implementers to get an XML file of the Core Component assembly instructions and associated semantic details, and therefore directly use them to transform actual physical business interchange content into the necessary format for them to be used in real documents/message interchanges.

Figure 2 below shows the adoption life cycle for core components and their use in such document assembly.
The initial version of the CCR has been designed with current available document standards in mind. These include DTD, XML Schema and RELAX variants. The aim in the future is to derive a standard CRI format based on XMI version 2. The reason for this is that the CRI is in a position to act as a transition point between the UML Tools world and the XML only world. Some issues preclude an early adoption of XMI, including enhanced UML tools that allow themselves to create classes based on meta-class models. The Core Component Meta Model defines a set of information that should be captured when new classes are generated. Clearly if a UML tool could enforce the capture of this information and then output it using the XMI format based on the Core Component Meta Model we would have a clear marriage between the two worlds of ebXML implementation specifics and the enhanced modelling concepts being developed for eBTWG.

In Section 2 below we examine in more detail the mechanics of deriving the metadata content as a series of logical steps. Figure 2 below provides a high-level schematic of the interactions associated with this process.

Figure 2. Overview of the adoption of Core Components Process



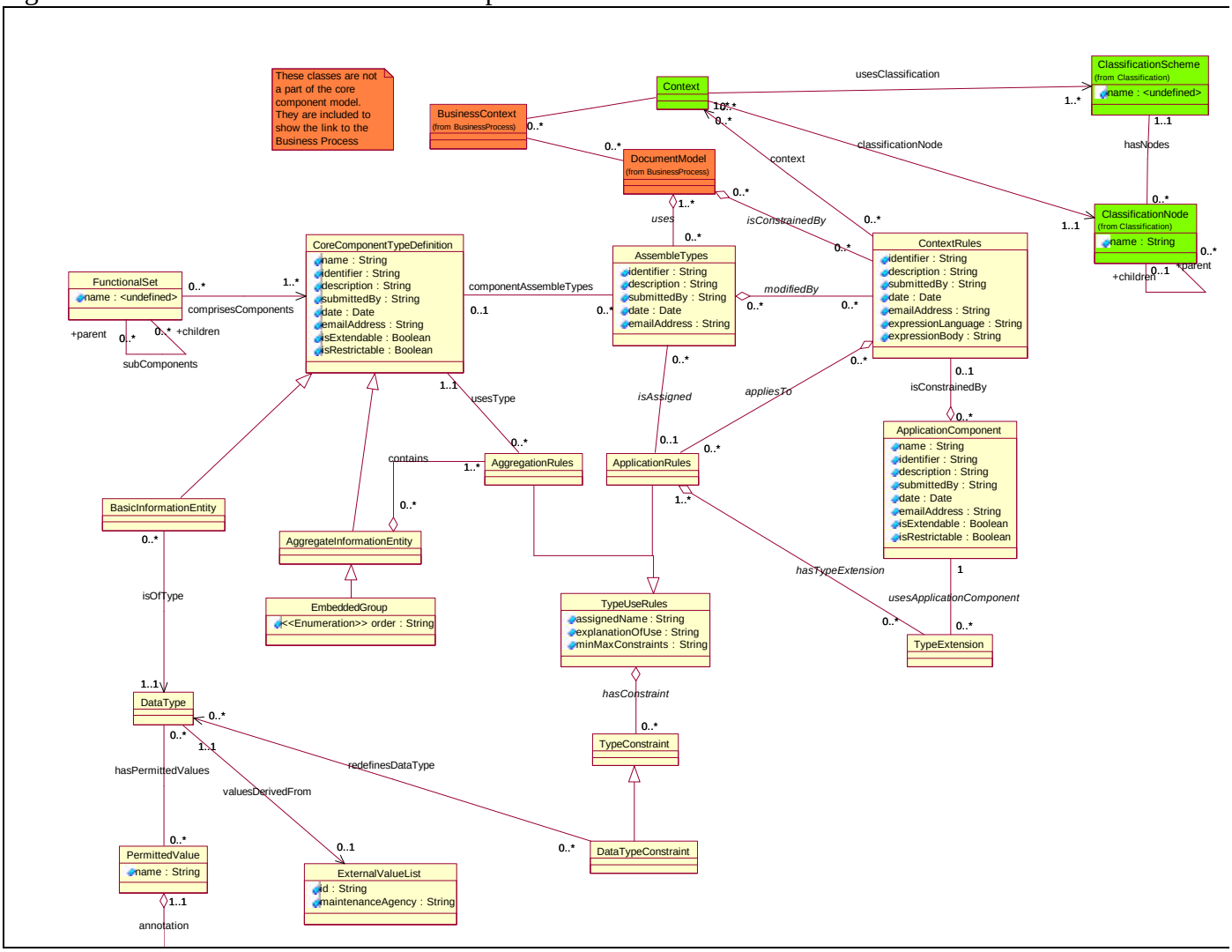
The above diagram explains the process by which both core components and legacy industry specific components get entered into the Registry via the CRI format. They are then linked to produce Business Information Entities (BIEs) that are then the semantic basis for what is passed across the wire in physical business interchange transactions. Starting in the bottom left quadrant we have UN Core Components (UNCC) and Components from other libraries either Industry verticals or Horizontal tool libraries. These are expressed in their native format. To be loaded into the registry they need to be

changed to the CRI format. This provides modellers and components builders the means to relate existing libraries to their UNCC equivalents to express real business components and their cross-walks between the industry implementations and the interoperable UNCC definitions. Associated with each of these core components is a reference key, a UID (Unique ID) reference value that allows it to be precisely addressed and versioned. The UID values therefore act as the network that defines the crosswalks and the interoperability matrix.

The underpinning for the CCR effort is the description of the XML document instance that carries the semantic information of each core component item. Figure 3 shows the UML information model of that representation.

The details of this information model, and the associated XML structure that holds the information is explained in the Section 2, the information model diagram is provided here to introduce the original concepts behind the early core components work so as to aid understanding of the continuing discussion in this section. The actually CRI work itself is a simplification and refined subset of this earlier work expressed as a simple XML instance structure.

Figure 3. Information Model of Core Components



2 Adopting and Implementing ebXML based systems.

Once core components have been established and made available then the CCR is designed to facilitate business organizations migrating their existing systems to ebXML and creating business artefacts in a systematic and deliberate set of steps. It also provides them with an extremely easy and rapid method of taking their in-place systems and making a baseline first step to adopting ebXML-based interoperability.

This section details the four steps or phases required and the outcomes and resources resulting to move from a start point today to a fully ebXML compatible implementation. These four phases are summarized here, and then each one is described in detail.

The ebXML CCR adoption steps for a particular industry domain are:

1) Relate existing legacy transaction formats to structural definitions containing UID references, and load and enhance the definitions of the UID items and structures into the ebXML Registry to complete as much of the CRI information as applicable.
2) Validate and migrate existing transactions and / or new transactions to conform to the best practices and XML representation guidelines and rules established by the ebXML specifications.
3) Participate in alignment efforts to relate the industry specific components to the broader UNCC definitions, and also related industry group work, including interoperability an alignment across industry. Use of UMM to facilitate this whole process.
4) Migrate existing interchange documents to reference and use UNCC UID references as substitutions for the older proprietary components. Migrate interchange documents to UN approved business process definitions and associated document payloads and ebXML enabled transportation and routing.

2.1 Step 1 – Relating existing legacy transaction formats

The first step is to take existing transactions being used in situ and associate with each of these a physical structure definition that relates UID references to each item within the transaction set. These structure definition may be DTD, XML Schema or RELAX or other machine parable structure syntax. Figure 4 below shows for a typical legacy piece of XML content (mailing Address) how to create a DTD example that makes the linkage to the semantic definitions via UID associations. We discuss later the rules for how UID values are created, essentially they consist of a character prefix followed by a 6 digit number as can be seen here such as 'CAT10100'.

Figure 4. Assigning UID values to legacy transaction structures

```
<!ELEMENT Address (Street+, City, (State | Province), (PostCode | ZIP),
                  Country?)>
<!ATTLIST Address UID CDATA #FIXED 'CAT10100'>
<!ELEMENT Street (#PCDATA)>
<!ATTLIST Street UID CDATA #FIXED 'CAT10101'>
<!ELEMENT City (#PCDATA)>
<!ATTLIST City UID CDATA #FIXED 'CAT10102'>
<!ELEMENT State (#PCDATA)>
<!ATTLIST State UID CDATA #FIXED 'CAT10103'>
<!ELEMENT Province (#PCDATA)>
<!ATTLIST Province UID CDATA #FIXED 'CAT10104'>
<!ELEMENT PostCode (#PCDATA)>
<!ATTLIST PostCode countrycode CDATA #REQUIRED
      UID CDATA #FIXED 'CAT10105'>
<!ELEMENT ZIP (#PCDATA)>
<!ATTLIST ZIP UID CDATA #FIXED 'CAT10106'>
<!ELEMENT Country (#PCDATA)>
<!ATTLIST Country UID CDATA #FIXED 'CAT10107'>
```

Then each of these UID references will point to entries in the ebXML Registry using the CRI XML instance containing the semantic definition of that individual item. See Section 2 for full details of what the CRI XML instance and associated semantics details being stored looks like, including some sample XML based off an OAG Address component example.

In assigning UID references an industry group, or individual business typically can chose a prefix string and number sequence that is appropriate for its own industry (as in the CAT10100 through to CAT10107 example above). There are no special rules that determine the number sequence or the prefix. However industry groups and companies may be expected to cooperate to share existing assigned UID values, and avoid collisions for new assignments. Also specifications for a central registry concept for ebXML are also under development as part of the phase 2 follow-on specification work.

For version specifics the ebXML Registry information model itself provides support for both major version and minor version assignments. This allows the base UID to always be

referenced, but with a version extension as needed, where the version details are a suffix separated by colons, i.e. CAT10107:01:00, is a major version reference.

2.2 Step 2 - Validate and migrate existing transactions

These actual best practice recommendations for ebXML based XML instances are explained elsewhere in the ebXML specifications. The major aspect that needs to be understood in regard to CCR is that the requirement here is not so much to micro-manage how content is assembled and represented in XML document instances. More important is that the overall principles and approach is adhered to. Specifically what needs to be avoided is any mechanisms that rely on specific tricks or proprietary mechanisms in mark-up that will inhibit or constrain interoperability based on the UID system and using the ebXML Registry as the means to provide the semantic pool and reference point.

The intent of the best practices is to suggest sensible techniques that will provide optimised, lightweight and simple structures that rely on well-defined and stable aspects of XML technology particularly. Obscure and esoteric techniques that rely on extended XML specifications and behaviours are to be expressly items that would be not favoured for inclusion in standard document formats. Such items lead to a reduced interoperability on a global scale by creating extended processing requirements on local implementations.

2.3 Step 3 – Participate in alignment and interoperability standardization

Once the industry domain or business organization has completed Step 1, then they have the underpinning necessary to begin the work of aligning with the existing UNCC base, and also alignment between related industry domain specifications. The eBTWG management structure contains specific working groups that provide resources to help in this process. Also the core components group have created documentation on how to discover, manage and develop core component definitions.

2.4 Step 4 – Migrate interchange documents to UN approved standards

Once the industry domain has established a base standard that is in alignment with ebXML, then the member base can migrate to those improved interchange formats.

2.5 Supplemental Notes on Semantic Alignment

A further goal of the core components work within ebXML is to provide standardized and uniformly named logical content. While this is attainable for logical components, this can be problematic for foreign languages especially in physical components.

To overcome this limitation the CRI structure within the model section provides for the ability to capture logical names. This has the additional benefit of freeing the XML element name from needing to conform to some artificial naming constraints within a physical implementation domain.

Some examples of an Address component taken from the Open Applications specifications have been provided as associated external document examples related to this physical text documentation herein.

3 XML Representation

This section describes the XML representation of Core Components. This section continues on from Part 1 and provides XML mechanisms for the model representations.
The representation is designed to support verbs as well as nouns, and also logical and well as physical models, and being able to associate business processes with core components. Furthermore the representation provides the means to capture context information, business rules and assembly information. The XML representation is the actual instance of the core component exposed in an XML structure and is therefore designed to facilitate application software mechanisms and use of core components throughout the ebXML technical architecture. Particularly important is the integration of the XML representation with the ebXML Registry information model and the ability to store the core component within an ebXML Registry and effectively manage and access it there.

Part 1 introduces the notion of basic core component, business component and document core component with each having an associated core component type. Then a core component is defined as "a building block for the creation of a semantically correct and meaningful information exchange 'parcel'". The XML representation therefore enables all of these mechanisms and in addition provides the contextual and process linkage for real world implementation.

To further the understanding of these mechanisms and the implementation here, this section is divided into three parts:

- The core component representation model,
- A summary of the main features and use cases,
- And then the actual schema of the XML with documentation of the components themselves and their intended purpose and content.

A sample core component noun instance is also provided in the addendum.

The intention is to provide a start point for implementers to be able to create core component noun instances themselves and the associated assembly instructions, and then to store those into an ebXML Registry system.

An important further note is that the XML representation of the core component is neutral to any particular schema dialect, whether it be DTD, XML Schema or RELAX, or some other variant such as EDI structures. The goal is to keep core components independent from any technology details, while allowing implementers to generate and choose whatever schema syntax best fits their business use. Furthermore this also provides future proofing against new schema syntax implementations and extensions. The XML representation does however allow implementers to embedded links to schema specific content as required by a specific schema dialect. An example

is a namespace or grammar link, or some complex piece of syntax fragment that assembly software will insert into a generated syntax instance.

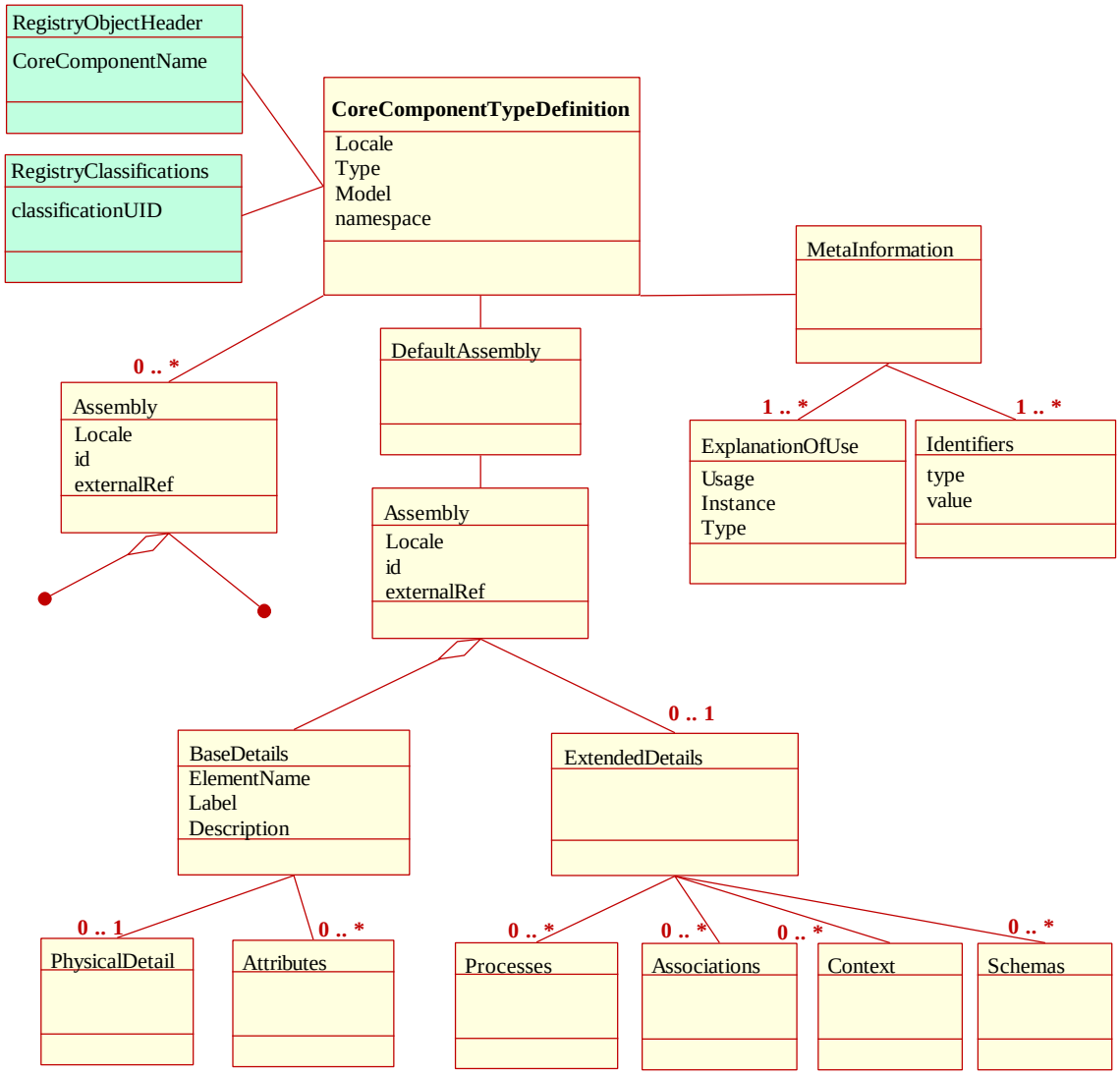
Therefore the schema representing the XML core component is documented in three ways: as a DTD, XML Schema and RELAX syntax structures. Each of these are interchangeable, and all describe the structure of the XML instance of the same core component, where this instance is intended to be as close to simple XML V1.1 syntax as possible.

To aid development use the latest instance of the core component schema and this documentation file will be available from an open source management library, with full versioning control. The CCR working group will be responsible for managing that library and posting latest changes to that.

3.1 Representation Model

The information model of the XML representation is shown in figure 5 and includes the registry object header content (see ebXML Registry specifications) that the ebXML registry header will associate with this core component item and contains important content details. It is important to remember that any registry object always contains these details and that the ebXML registry services provide many query and access tools keyed off this content along with audit, owner, security and tracking mechanisms.

Figure 5. Core component instance information model



This model diagram has been organized to show the major features for clarity, expanded down to only the 3rd level of nodes. Similarly, properties within objects have been restricted to skeletal notes only. The actual schema instance model is shown next to detail the complete entities that are represented beneath in the full model. Also, no attempt has been made to show relation of use cases to model content; for instance, a logical model core component may not contain physical detail content, only associations to physical model core components as assembly details (see section 2 below for use case discussion). Similarly a verb entry or a process entry may have restricted physical detail items, and so on.

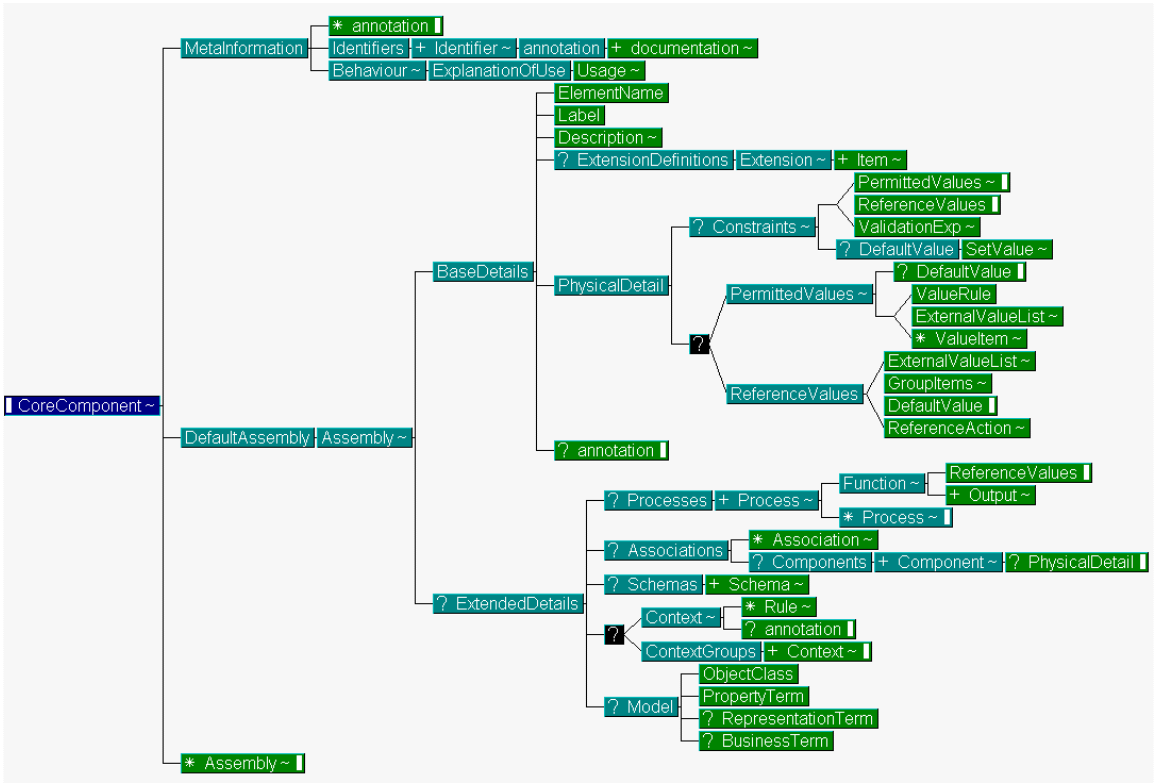
Part 2 – Implementation and Adoption

Methods, Technical Details, Approach

4 Implementation Diagrams

The XML schema model of the XML representation is shown in figure 5.1.0 includes only the core component itself, (without the registry information model items). Only the element level model is shown, not the attribute level. The details of the attributes are given in major section below detailing the documentation of the schema itself. (Note: elements have been labelled using UN spellings, not North American spellings)

Figure 5.1.0. XML schema model (DTD representation)



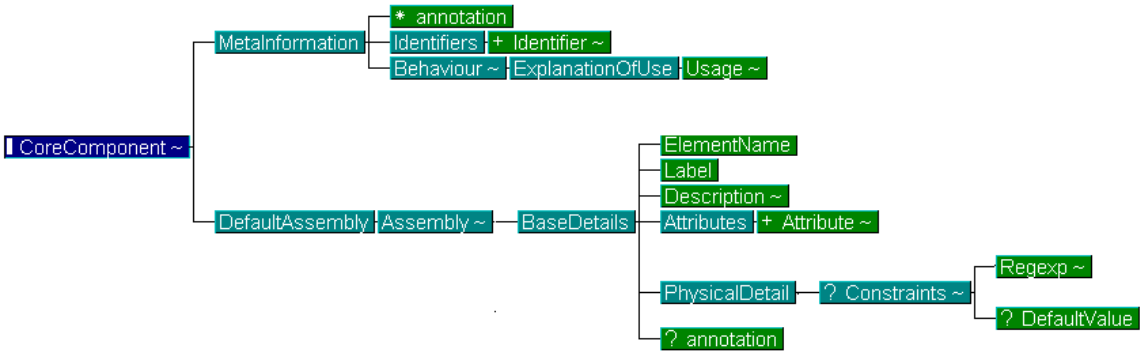
The figure shows the complete hierarchy of the XML core component instance. It is also designed to provide logical and consistent use of XML markup, to facilitate XPath based selections against content, and particularly to facilitate accessing such content within the ebXML Registry. In this regard a DefaultAssembly is always provided, so that consistent access can be made against reliable content for all types and aspects of core components themselves. The overall design allows for use in representing the multiple aspects and types of core components.

In the next section, 5.1 use case diagrams showing selected instances of the complete model will be provided, along with selected uses.

4.1 Features and Use Cases

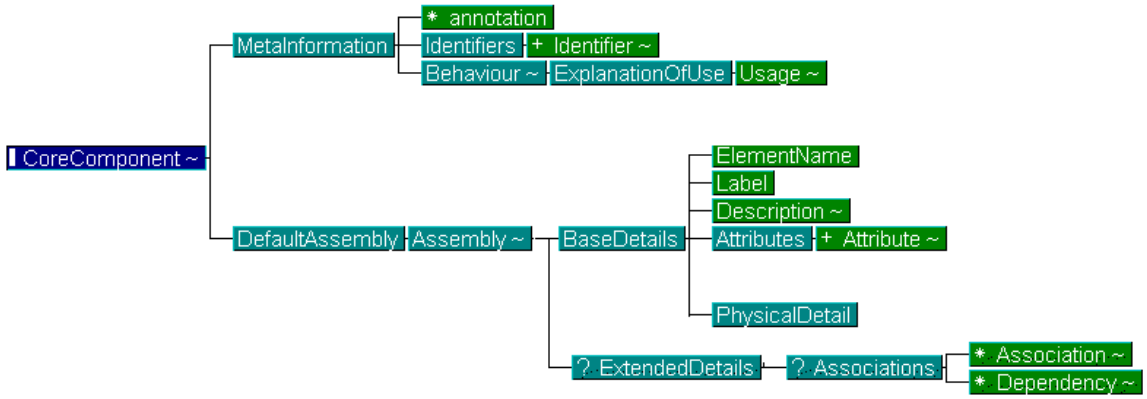
This section presents some obvious and primary use cases for the XML representation of a core component. These include a simple noun (atomic core component), a logical core component, and a permitted values list core component. Then an assembly core component and a business process (verb) core component. Each of these samplings is shown as a schematic representation, where the required elements are included in the view, and those optional or not required items are removed, or collapsed.

Figure 5.1.1. Atomic core component (noun)



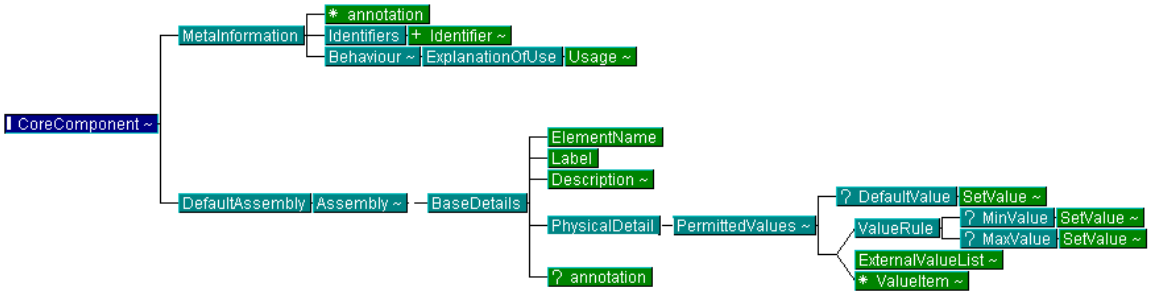
The figure shows those items that typical for this kind of core component. Other core component assemblies would likely reference this one using a UID address.

Figure 5.2.1. Logical core component (noun)



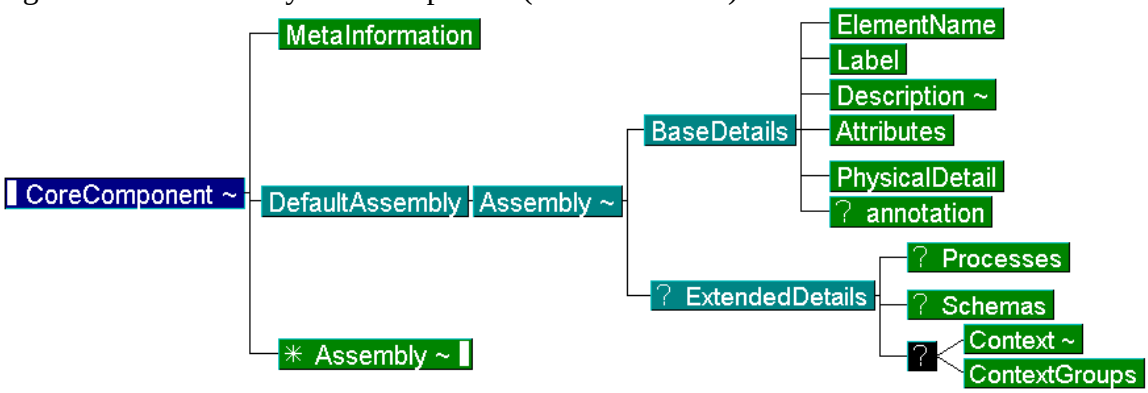
The figure shows those items that typical for this kind of core component. This would like reference other core component assemblies or atomic core components using their UID address(es).

Figure 5.3.1. Permitted values core component (noun)



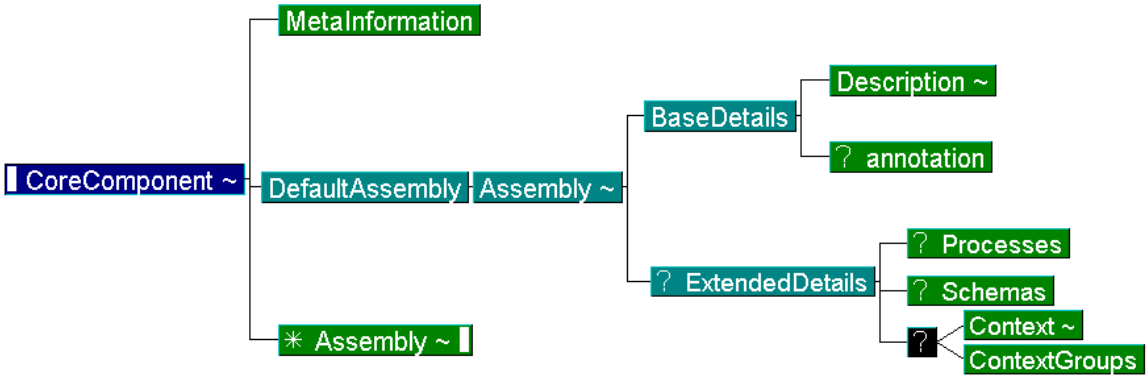
The figure shows those items that typical for this kind of core component.

Figure 5.4.1. Assembly core component (noun collection)



The figure shows those items that typical for this kind of core component. The Assembly indicates the structure and included atomic core components and how they are arranged together. Typically this type of core component is a business document for exchanging in a business process.

Figure 5.5.1. Business process core component (verb)



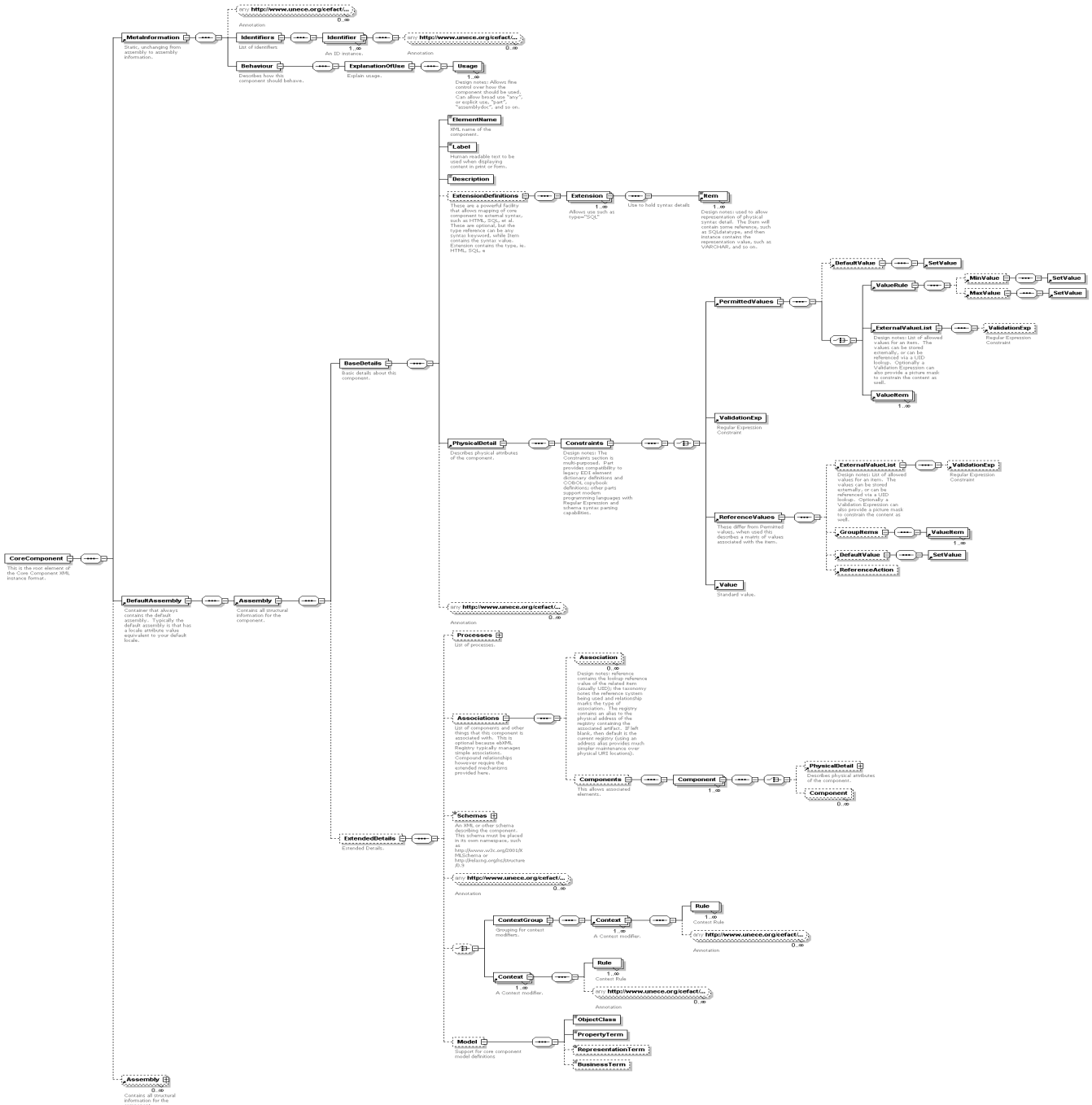
The figure shows those items that typical for this kind of core component.

4.2 Core Component Schema

This section contains the reference schema for the XML implementation. Three schema flavours are provided: DTD, RELAX and XML-Schema. Each one is equivalent and can be used interchangeably. The DTD representation is used as the reference one. The various formats will not be maintained in this document. An automatically generated glossary made from the XSD schema is presented. The actual DTD, XML-Schema and RELAX files are contained in the zip associated with this document.

The diagram below, in Figure 6, is an overview of the XML-Schema version of the CRI.

492 Figure 6. XML Schema Overview Diagram
493



494
495

4.3 Core Component Schema Glossary Details

The Glossary is displayed in two formats

- Main Parent Elements

And

- Alphabetically by main elements and sub-elements (all other child elements).

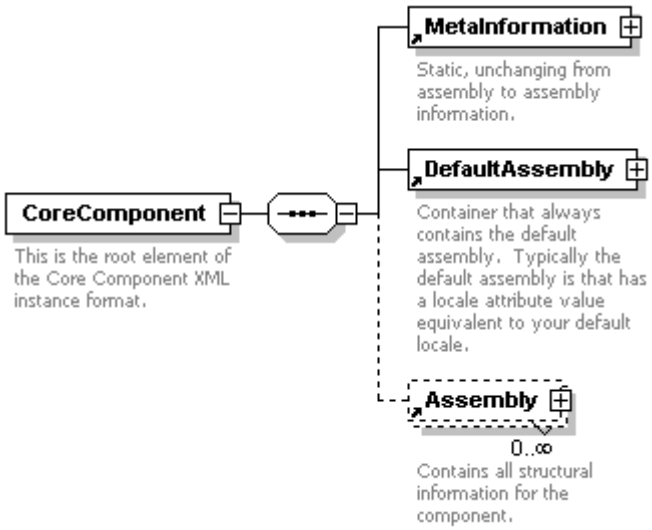
Each item is annotated to describe the specific individual function. It should be noted that the schema is designed to fulfill a flexible set of uses, including:

- Logical core components
- Physical component details
- Assembly documents
- Compound items
- Atomic nouns
- Atomic and compound parts of nouns
- Process Verbs
- Reference tables

As a guide for implementers, users should be encouraged to first create assembly document references based off their existing DTD, Schema or EDI transaction structures. From this naturally flows the associated component details, and then above those the logical core component models can be derived. The associated noun and part details can then be derived along with verb and reference table details.

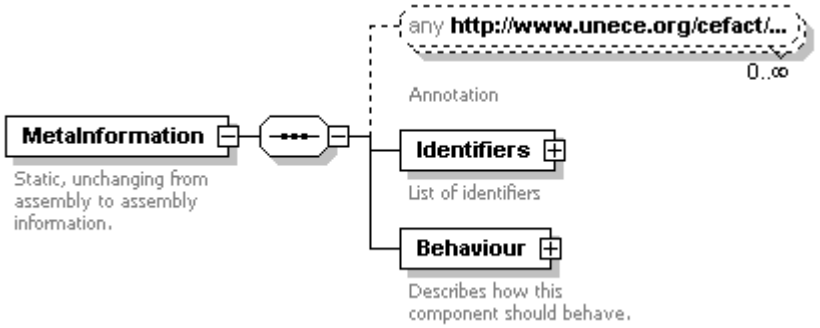
516

517 element **CoreComponent**

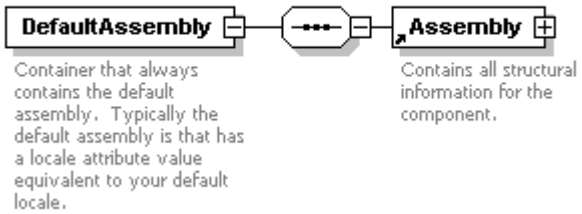
diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**MetaInformation DefaultAssembly Assembly**
attributes	Name / Type / Use / Default / Fixed defaultLocale / locales / optional / en_US type / xs:NMTOKEN / required model / xs:string / optional / physical
annotation	documentation documentation documentation This is the root element of the Core Component XML instance format.$Header: /cvsroot/corecomponents/schemas/CoreComponent.xsd,v 1.17 2001/10/07 18:33:34 matt Exp $Design notes: The settings of the type and model attributes determine the particular flavour of core component represented. From physical or logical model, to atomic element or compound assembly, an extensive range of permutations is supported to meet all the essential baseline representations for e-business artifacts. The MetaInformation is required for all core components and describes the Identifiers and Behaviour of the core component. The Identifiers mechanism is intended to capture primarily UID references, or alternatively UDDI or can be extended for future reference systems. (Note: a UID reference can contain versioning information as a suffix). The annotation is provided for compatibility with XSchema annotations. The Behaviour element determines if the core component can be inherited and changed. Then the Usage element attributes are set to indicate the type of use for the core component.

518
519
520
521
522
523
524
525

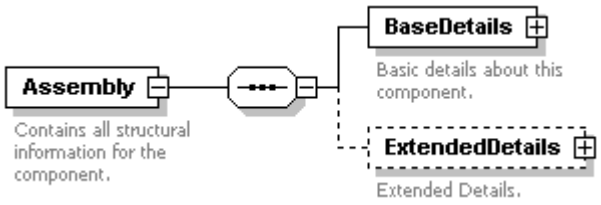
526
527 element **MetaInformation**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Identifiers** **Behaviour**
used by	element **CoreComponent**
annotation	documentation Static, unchanging from assembly to assembly information.

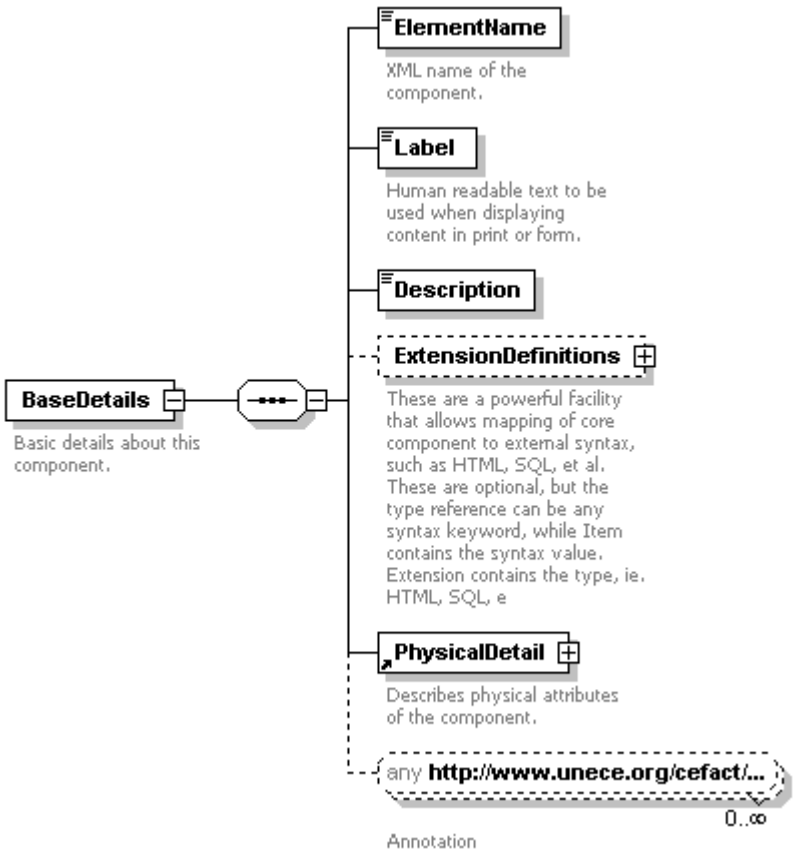
528
529 element **DefaultAssembly**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
children	**Assembly**				
used by	element **CoreComponent**				
attributes	Name ref	Type xs:string	Use optional	Default	Fixed
annotation	documentation documentation	Container that always contains the default assembly. Typically the default assembly is that has a locale attribute value equivalent to your default locale.Additional design notes: The DefaultAssembly references Assembly, and ensures that there is always a primary set of information with which the core component can be referenced, regardless of whether the core component has an additional Assembly or not. The additional Assembly is specifically to provide locale information for other languages in addition to the defaultlocale defined on the CoreComponent element, as well as alternate assembly details			

element **Assembly**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**BaseDetails** **ExtendedDetails**
used by	elements **CoreComponent** **DefaultAssembly**
attributes	Name / Type / Use / Default / Fixed locale / locales / required id / xs:string / optional externalRef / xs:anyURI / optional
annotation	documentation Contains all structural information for the component.

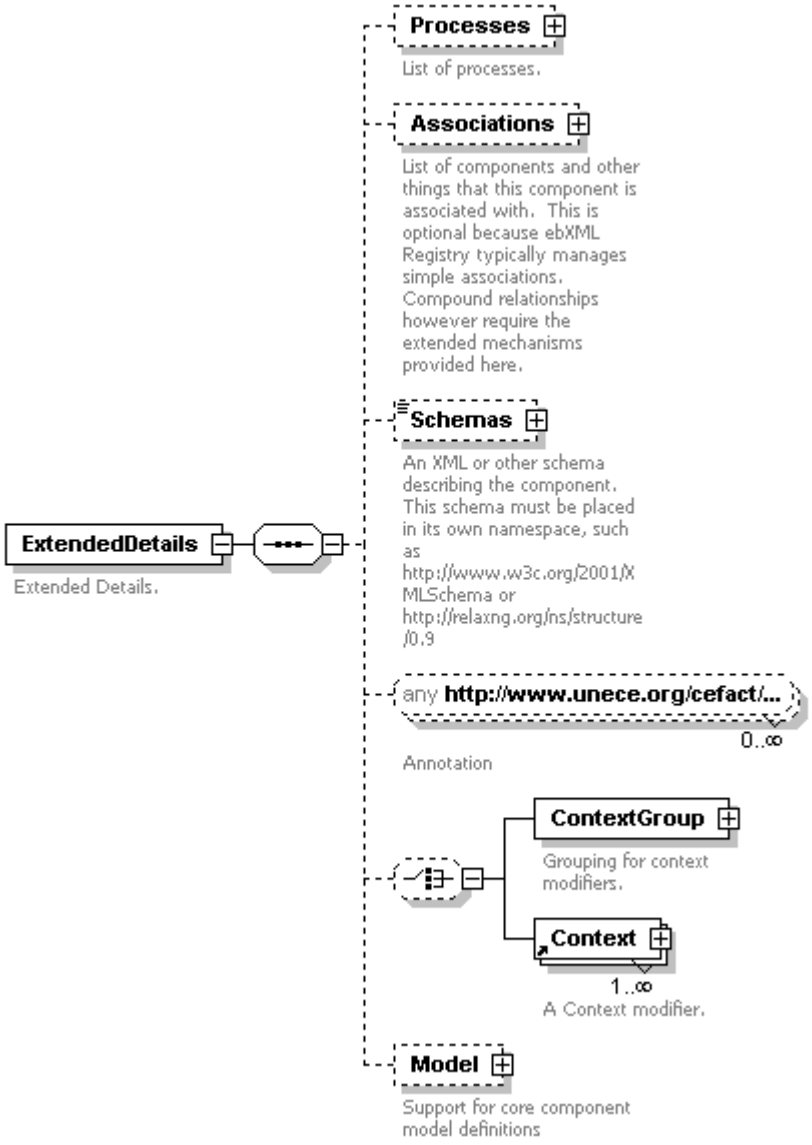
element **Assembly/BaseDetails**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0

children	**ElementName Label Description ExtensionDefinitions PhysicalDetail**	
annotation	documentation documentation	Basic details about this component.The BaseDetails capture the primitive information about a core component. The ElementName is either the atomic XML tagname (for the default locale) or root tag name for compound or complex items. For permitted values lists, the ElementName similarly points to the default associated element, or may simply be EMPTY if not applicable. Label is the human readable text to be displayed on a form or printed on a report associated with this core component, and again the default locale language applies. The Description is a short text documentation of the core component, while attributes on Description allow referencing to extended content fully documenting the item. The Attributes equate exactly to XML markup attributes and solve the problem of referencing attributes of attributes within a message instance. The Attributes element block allows attributes to be detailed in-line, or by referencing to another UID address where extended information about the attribute can be referenced.

536

537 element **Assembly/ExtendedDetails**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Processes Link018B5C30 Schemas ContextGroup Context Model**

annotation	documentation documentation	Extended Details.The ExtendedDetails group is a set of optional items providing advanced information about the core component. These are optional and therefore are intended for different types of core component to handle the extended details they may require. For example the Associations provide the ability to provide a mapping crosswalk of equivalent or similar items within an industry and across different industry domains.

538
539
540 Schema **CoreComponent.xsd**
541
542

targetNamespace: **http://www.unece.org/cefact/cri/1.0**

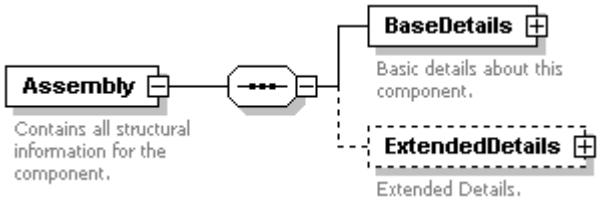
543

Elements	Simple types
Assembly	**baseDataTypes**
Context	**contextTypes**
CoreComponent	**locales**
DefaultAssembly	**regexpTypes**
DefaultValue	
ExternalValueList	
Function	
GroupItems	
Item	
MapValue	
MaxValue	
MetaInformation	
MinValue	
PermittedValues	
PhysicalDetail	
Process	
ReferenceAction	
ReferenceValues	
SetValue	
ValidationExp	
Value	
ValueItem	
ValueRule	

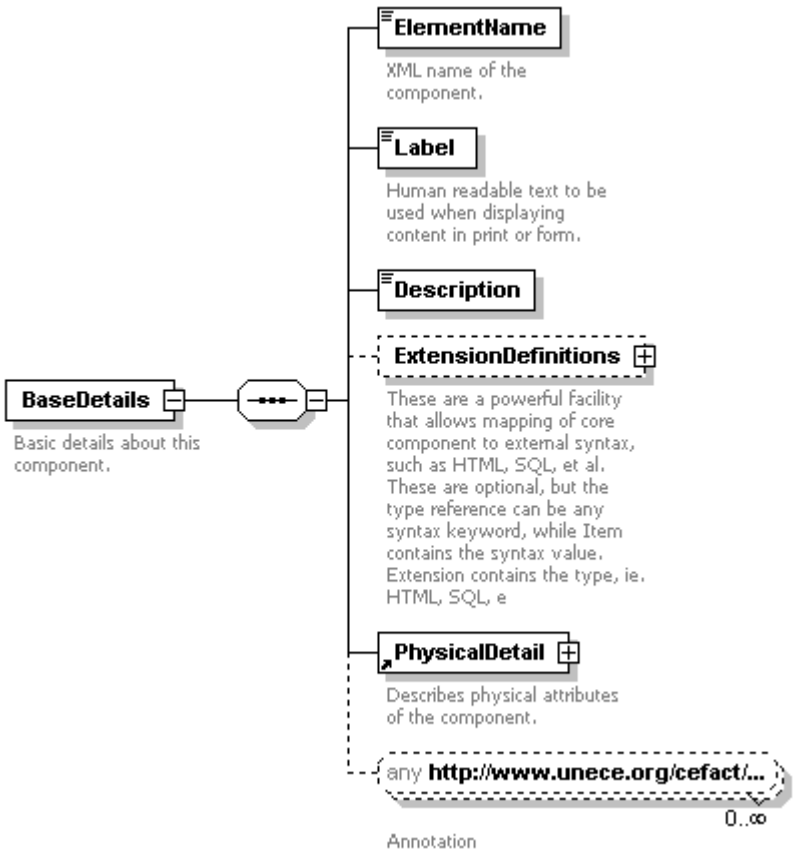
544
545

4.4 Alphabetical Listing of Structure Components.

element **Assembly**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	BaseDetails ExtendedDetails
used by	elements CoreComponent DefaultAssembly
attributes	Name: locale; Type: locales; Use: required; Default: ; Fixed: Name: id; Type: xs:string; Use: optional; Default: ; Fixed: Name: externalRef; Type: xs:anyURI; Use: optional; Default: ; Fixed:
annotation	documentation: Contains all structural information for the component.

element **Assembly/BaseDetails**

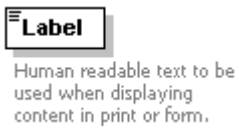
diagram	

namespace	http://www.unece.org/cefact/cri/1.0	
children	**ElementName Label Description ExtensionDefinitions PhysicalDetail**	
annotation	documentation documentation	Basic details about this component.The BaseDetails capture the primitive information about a core component. The ElementName is either the atomic XML tagname (for the default locale) or root tag name for compound or complex items. For permitted values lists, the ElementName similarly points to the default associated element, or may simply be EMPTY if not applicable. Label is the human readable text to be displayed on a form or printed on a report associated with this core component, and again the default locale language applies. The Description is a short text documentation of the core component, while attributes on Description allow referencing to extended content fully documenting the item. The Attributes equate exactly to XML markup attributes and solve the problem of referencing attributes of attributes within a message instance. The Attributes element block allows attributes to be detailed in-line, or by referencing to another UID address where extended information about the attribute can be referenced.

551
552
553 element **Assembly/BaseDetails/ElementName**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
type	**xs:string**	
annotation	documentation documentation	XML name of the component.Design notes: A default name to be used for an XML tag. This is not a mandated name. Since the UID provides an independent identifier implementers are free to use alternate tag label names as needed. However this name is a useful reference and should be used if a local implementation does not already have an equivalent. Typically this entry should also be a valid XML name, but should not be assumed to be (i.e. transformation tools should validate for white space, invalid characters and replace accordingly as required by the specific application).

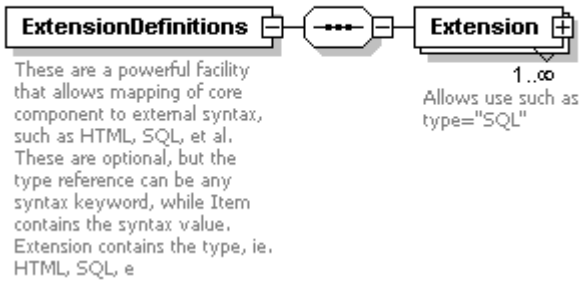
554
555
556 element **Assembly/BaseDetails/Label**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
type	**xs:string**	
annotation	documentation documentation	Human readable text to be used when displaying content in print or form.Design notes: Human readable label. Language text should correspond to the locale setting.

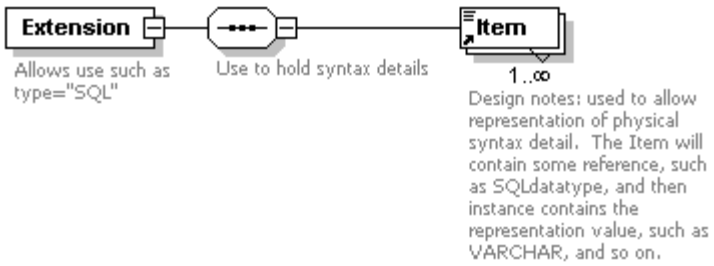
557
558
559 element **Assembly/BaseDetails/Description**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
type	**xs:string**	

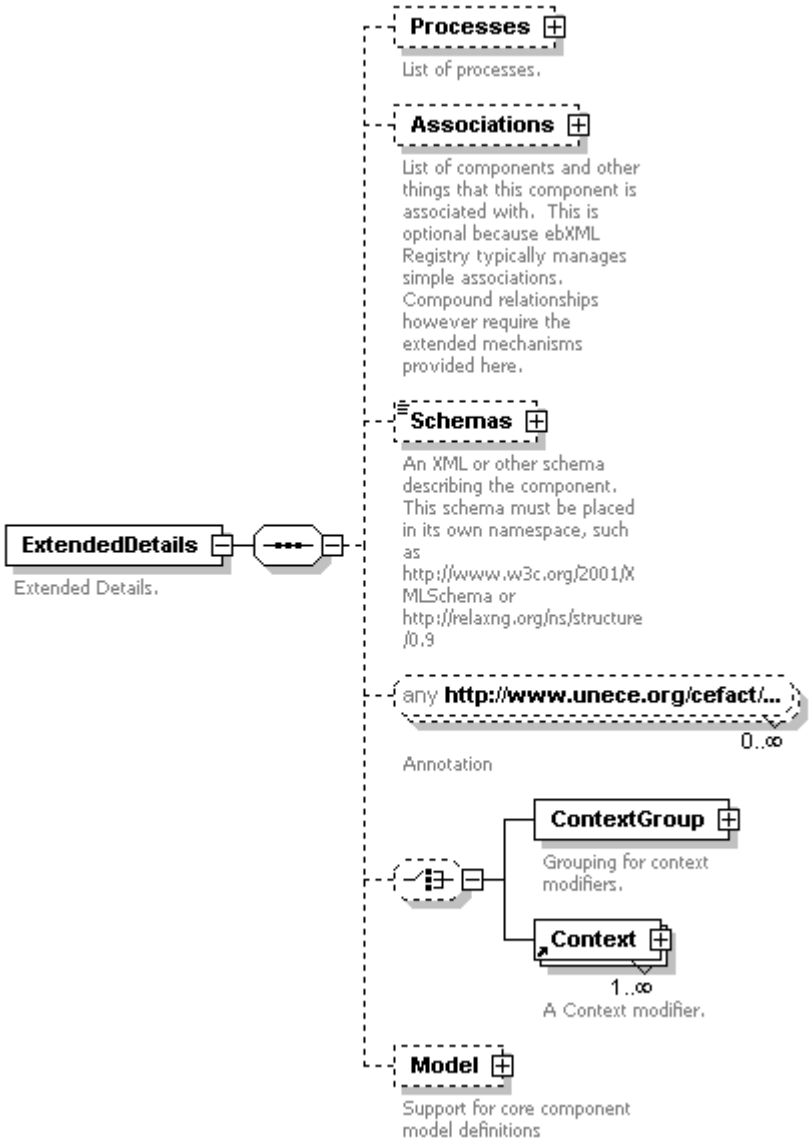
560
561
562 element **Assembly/BaseDetails/ExtensionDefinitions**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Extension**
annotation	documentation documentation: These are a powerful facility that allows mapping of core component to external syntax, such as HTML, SQL, et al. These are optional, but the type reference can be any syntax keyword, while Item contains the syntax value. Extension contains the type, ie. HTML, SQL, eDesign notes: This mechanism is a catchall. Implementers will require their own specific local extensions. This system provides this, and allows any physical syntax detail to be represented and retrieved. An example is a SQL representation of a particular physical component noun.

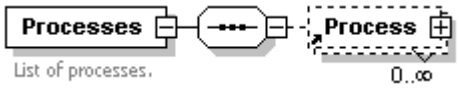
563
564
565 element **Assembly/BaseDetails/ExtensionDefinitions/Extension**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Item**
attributes	Name: name — Type: xs:string — Use: required Name: type — Type: xs:string — Use: required
annotation	documentation documentation: Allows use such as type="SQL"Design notes: used to allow representation of physical syntax detail as a programming support device, where programmers require explicit ability to capture syntax related details. The type is the physical type (such as SQL or XForm) of the particular syntax. The name is the reference value to this item. Typically this will be the UID with a suffix to denote its use, such as OAG023000:SQL that can therefore be directly referenced in an XPath or similar lookup.

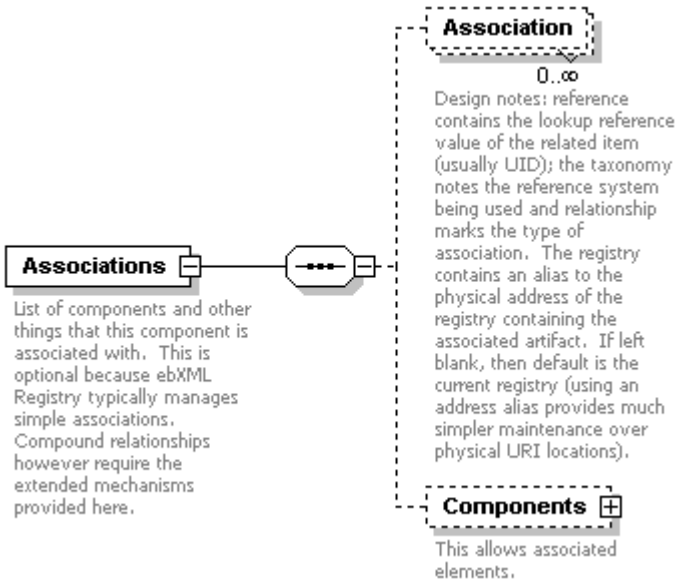
566
567
568 element **Assembly/ExtendedDetails**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	Processes Associations Schemas ContextGroup Context Model
annotation	documentation documentation: Extended Details.The ExtendedDetails group is a set of optional items providing advanced information about the core component. These are optional and therefore are intended for different types of core component to handle the extended details they may require. For example the Associations provide the ability to provide a mapping crosswalk of equivalent or similar items within an industry and across different industry domains.

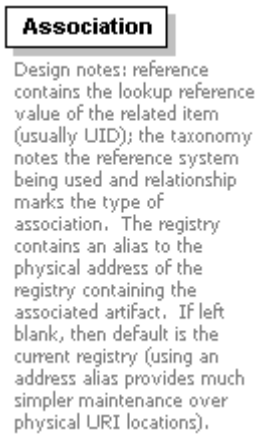
569
570
571 element **Assembly/ExtendedDetails/Processes**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
children	**Process**	
annotation	documentation documentation	List of processes. Design notes: provides linkage to BPSS via a simple mechanism to optionally capture the actual process steps, and then link the formal BPSS definitions of those to the steps. Can be used to either declare a component verb, or to associate a process to a core component, such as a transaction, or vice versa. (This part of the CRI will be refined in collaboration with the BPSS working group).

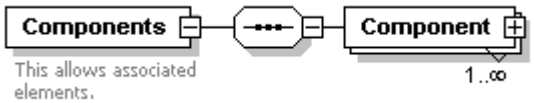
572
573
574 element **Assembly/ExtendedDetails/Associations**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
children	**Association Components**	
annotation	documentation	List of components and other things that this component is associated with. This is optional because ebXML Registry typically manages simple associations. Compound relationships however require the extended mechanisms provided here.

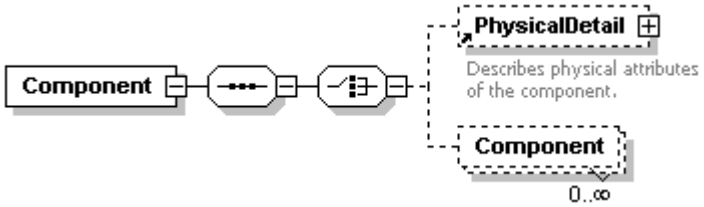
element **Assembly/ExtendedDetails/Associations/Association**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
attributes	Name / Type / Use / Default / Fixed reference / xs:string / required taxonomy / xs:string / optional relationship / xs:NMTOKEN / required registry / xs:string / optional
annotation	documentation: Design notes: reference contains the lookup reference value of the related item (usually UID); the taxonomy notes the reference system being used and relationship marks the type of association. The registry contains an alias to the physical address of the registry containing the associated artifact. If left blank, then default is the current registry (using an address alias provides much simpler maintenance over physical URI locations).

element **Assembly/ExtendedDetails/Associations/Components**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Component**
annotation	documentation documentation documentation documentation: This allows associated elements.Example: ZIP code requires City and State as required fields, and Country as optional.This also supports modelling tools thru the Direction attribute.Design notes: The Dependency differs from Associations in that this lays down items that belong together, such as ZIP code, City, State, Address and determines optional or required relations. Also for modelling tools, the Direction is included.

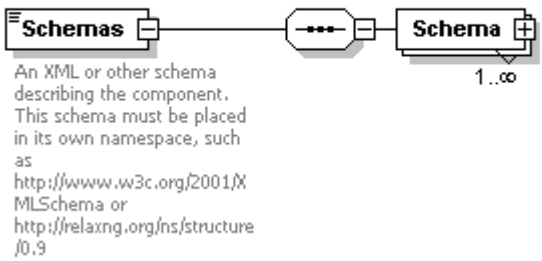
element **Assembly/ExtendedDetails/Associations/Components/Component**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	PhysicalDetail

attributes

Name	Type	Use	Default	Fixed
name	xs:string	required		
objType	xs:string	optional		
objMode	xs:string	optional		
use	xs:NMTOKEN	required		
relation	xs:NMTOKEN	required		
direction	xs:NMTOKEN	required		
UIDreference	xs:string	optional		
taxonomy	xs:NMTOKEN	optional		
registry	xs:string	optional		
note	xs:string	optional		

element **Assembly/ExtendedDetails/Schemas**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	Schema

attributes	Name	Type	Use	Default	Fixed
	type				

annotation		
	documentation	An XML or other schema describing the component. This schema must be placed in its own namespace, such as http://www.w3c.org/2001/XML-Schema or http://relaxng.org/ns/structure/0.9

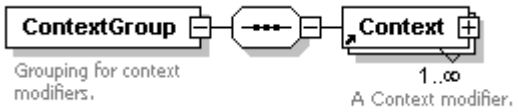
element **Assembly/ExtendedDetails/Schemas/Schema**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0

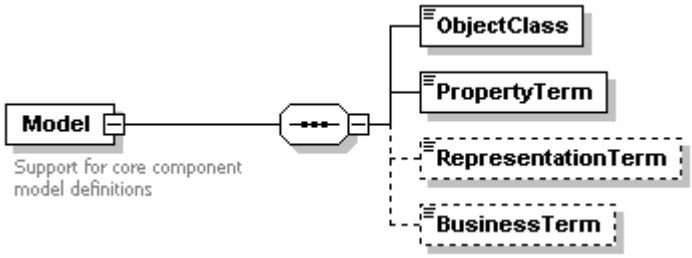
attributes	Name	Type	Use	Default	Fixed

	location	xs:anyURI	optional	
	type	xs:string	required	W3C

element **Assembly/ExtendedDetails/ContextGroup**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
children	**Context**	
annotation	documentation documentation	Grouping for context modifiers.Design notes: Context reference mechanisms are still under development by the modelling working group members. The mechanism here is a start point and provides basic functionality that will doubtless be refined for later specification releases.

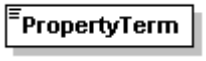
element **Assembly/ExtendedDetails/Model**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
children	**ObjectClass PropertyTerm RepresentationTerm BusinessTerm**	
annotation	documentation documentation	Support for core component model definitionsDesign notes: This section captures the logical model details from the core component discovery working group results and makes it available in a consistent XML form directed by the registry management services, tracking and controls. See the core components specifications for more details of the catalogue provided.

element **Assembly/ExtendedDetails/Model/ObjectClass**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
type	**xs:string**

element **Assembly/ExtendedDetails/Model/PropertyTerm**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
type	**xs:string**

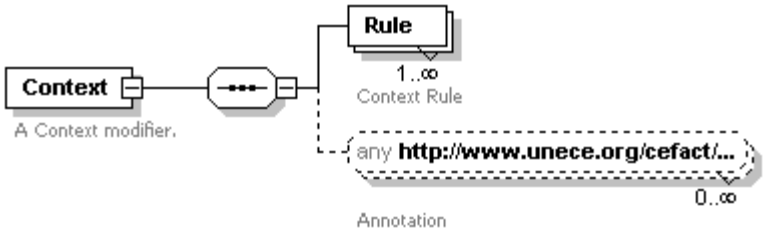
602
603
604 element **Assembly/ExtendedDetails/Model/RepresentationTerm**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
type	**xs:string**

605
606
607 element **Assembly/ExtendedDetails/Model/BusinessTerm**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
type	**xs:string**

608
609
610 element **Context**

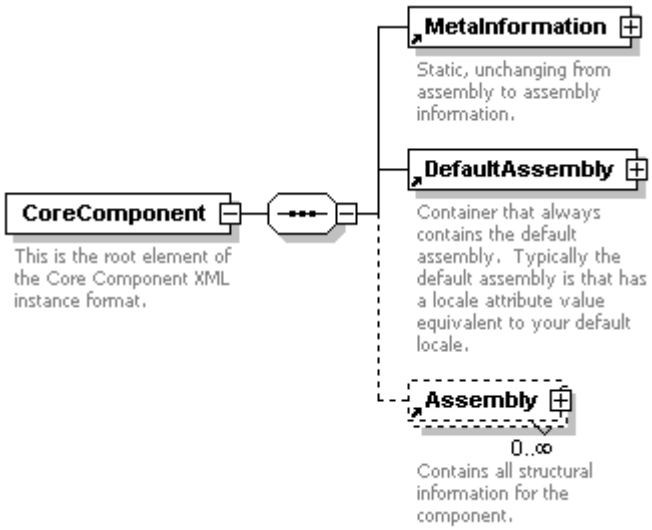
diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	Rule
used by	elements Assembly/ExtendedDetails/ContextGroup Assembly/ExtendedDetails
attributes	Name / Type / Use / Default / Fixed locale / locales / required id / xs:string / optional
annotation	documentation A Context modifier.

611
612
613 element **Context/Rule**

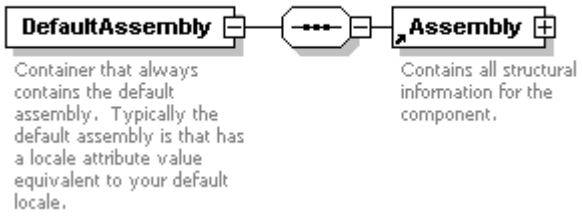
diagram	
namespace	http://www.unece.org/cefact/cri/1.0
attributes	Name / Type / Use / Default / Fixed type / contextTypes / required name / xs:string / required value / xs:string / required classificationSchem e / xs:string / optional label / xs:string / optional
annotation	documentation documentation Context Rule Design notes: Basic mechanisms for capturing rule details. Implementers should note however that no formal rule syntax is intended and that

	individual vendors will likely support their own mechanisms first before consistent methods are standardized.

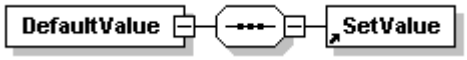
614
615
616 element **CoreComponent**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**MetaInformation** **DefaultAssembly** **Assembly**
attributes	Name / Type / Use / Default / Fixed defaultLocale / locales / optional / en_US type / xs:NMTOKEN / required model / xs:string / optional / physical
annotation	documentation documentation documentation This is the root element of the Core Component XML instance format.$Header: /cvsroot/corecomponents/schemas/CoreComponent.xsd,v 1.17 2001/10/07 18:33:34 matt Exp $Design notes: The settings of the type and model attributes determine the particular flavour of core component represented. From physical or logical model, to atomic element or compound assembly, an extensive range of permutations is supported to meet all the essential baseline representations for e-business artifacts. The MetaInformation is required for all core components and describes the Identifiers and Behaviour of the core component. The Identifiers mechanism is intended to capture primarily UID references, or alternatively UDDI or can be extended for future reference systems. (Note: a UID reference can contain versioning information as a suffix). The annotation is provided for compatibility with XSchema annotations. The Behaviour element determines if the core component can be inherited and changed. Then the Usage element attributes are set to indicate the type of use for the core component.

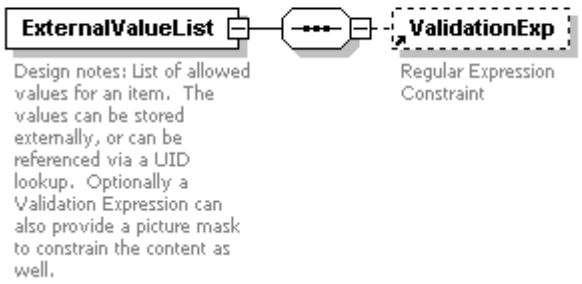
element **DefaultAssembly**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Assembly**
used by	element **CoreComponent**
attributes	Name ref; Type xs:string; Use optional; Default; Fixed
annotation	documentation documentation: Container that always contains the default assembly. Typically the default assembly is that has a locale attribute value equivalent to your default locale.Additional design notes: The DefaultAssembly references Assembly, and ensures that there is always a primary set of information with which the core component can be referenced, regardless of whether the core component has an additional Assembly or not. The additional Assembly is specifically to provide locale information for other languages in addition to the defaultlocale defined on the CoreComponent element, as well as alternate assembly details

element **DefaultValue**

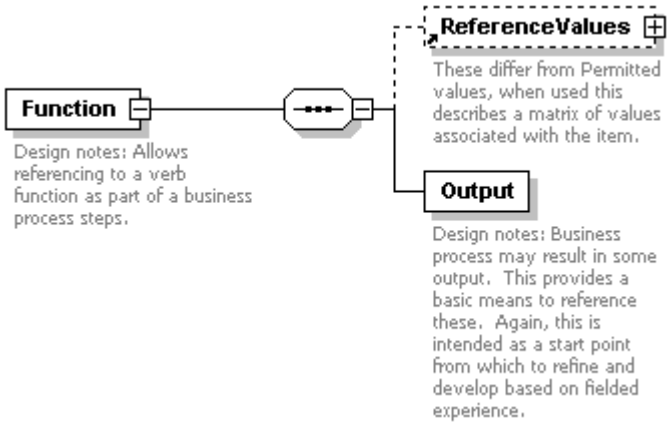
diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**SetValue**
used by	elements **PermittedValues** **ReferenceValues**

element **ExternalValueList**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0

children	ValidationExp				
used by	elements PermittedValues ReferenceValues				
attributes	Name	Type	Use	Default	Fixed
	reference	xs:string	required		
	taxonomy	xs:NMTOKEN	required		
	registry	xs:string			
annotation	documentation	Design notes: List of allowed values for an item. The values can be stored externally, or can be referenced via a UID lookup. Optionally a Validation Expression can also provide a picture mask to constrain the content as well.			

element **Function**

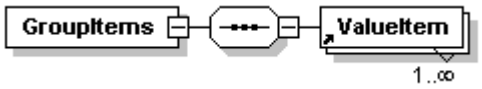
diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
children	ReferenceValues Output				
used by	element Process				
attributes	Name	Type	Use	Default	Fixed
	itemName	xs:string	optional		
	itemReference	xs:string	optional		
	UIDReference	xs:string	optional		
annotation	documentation	Design notes: Allows referencing to a verb function as part of a business process steps.			

element **Function/Output**

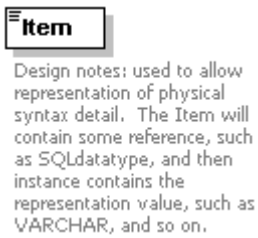
diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
attributes	Name	Type	Use	Default	Fixed
	businessRule	xs:string	optional		
	itemName	xs:string	optional		
	processClassUID	xs:string	optional		
	functionClassUID	xs:string	optional		

annotation	documentation	Design notes: Business process may result in some output. This provides a basic means to reference these. Again, this is intended as a start point from which to refine and develop based on fielded experience.

element **GroupItems**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
children	**ValueItem**				
used by	element **ReferenceValues**				
attributes	Name comment	Type xs:string	Use	Default	Fixed

element **Item**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
type	restriction of **xs:string**	
used by	element **Assembly/BaseDetails/ExtensionDefinitions/Extension**	
annotation	documentation	Design notes: used to allow representation of physical syntax detail. The Item will contain some reference, such as SQLdatatype, and then instance contains the representation value, such as VARCHAR, and so on.

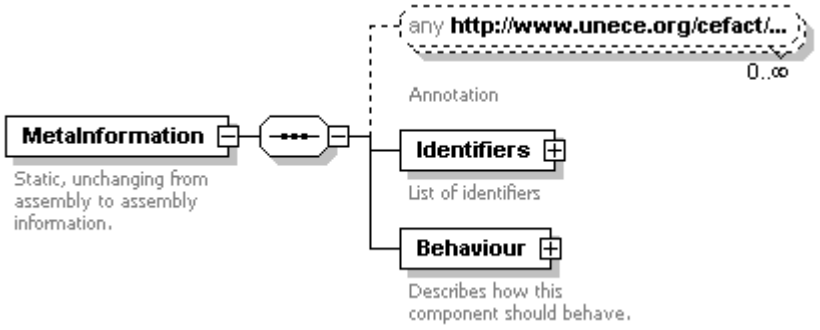
element **MapValue**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
type	extension of **xs:string**				
attributes	Name key default	Type xs:string xs:boolean	Use required optional	Default	Fixed
annotation	documentation	A key=value pair.			

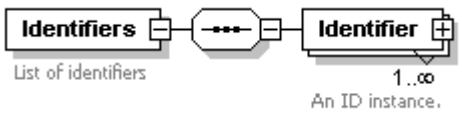
641
642
643 element **MaxValue**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**SetValue**
used by	element **ValueRule**

644
645
646 element **MetaInformation**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Identifiers** **Behaviour**
used by	element **CoreComponent**
annotation	documentation Static, unchanging from assembly to assembly information.

647
648
649 element **MetaInformation/Identifiers**

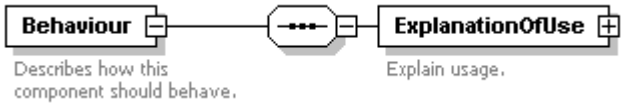
diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**Identifier**
annotation	documentation List of identifiers

650
651
652 element **MetaInformation/Identifiers/Identifier**

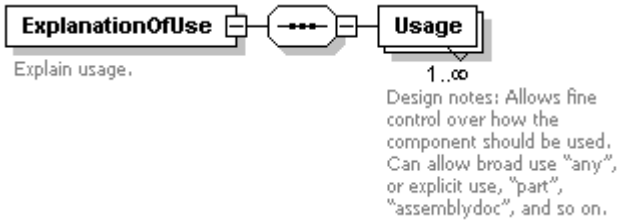
diagram	
namespace	http://www.unece.org/cefact/cri/1.0

attributes	Name	Type	Use	Default	Fixed
	type	xs:string	required		
	value	xs:string	required		
annotation	documentation	An ID instance.			

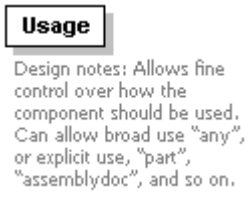
653
654
655 element **MetaInformation/Behaviour**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
children	ExplanationOfUse				
attributes	Name	Type	Use	Default	Fixed
	isRestrictable	xs:boolean	optional	false	
	isExtendable	xs:boolean	optional	true	
annotation	documentation	Describes how this component should behave.			

656
657
658 element **MetaInformation/Behaviour/ExplanationOfUse**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
children	Usage				
annotation	documentation	Explain usage.			

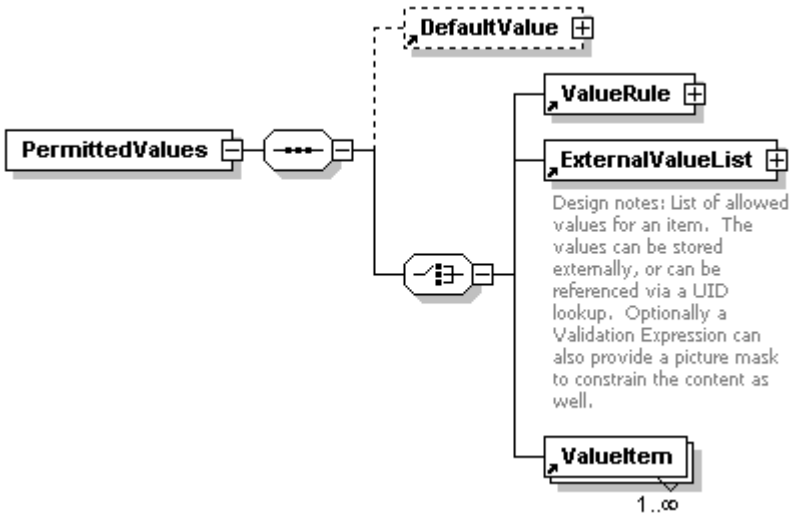
659
660
661 element **MetaInformation/Behaviour/ExplanationOfUse/Usage**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
attributes	Name	Type	Use	Default	Fixed
	type	xs:NMTOKEN	required		
	instance	xs:NMTOKEN	required		
annotation	documentation	Design notes: Allows fine control over how the component should be used. Can allow broad use "any", or explicit use, "part", "assemblydoc", and so on.			

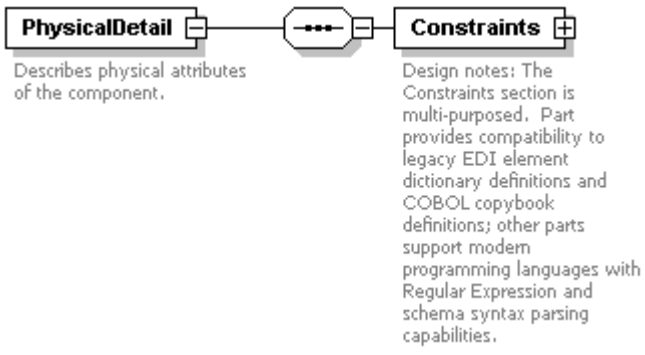
element **MinValue**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	SetValue
used by	element ValueRule

element **PermittedValues**

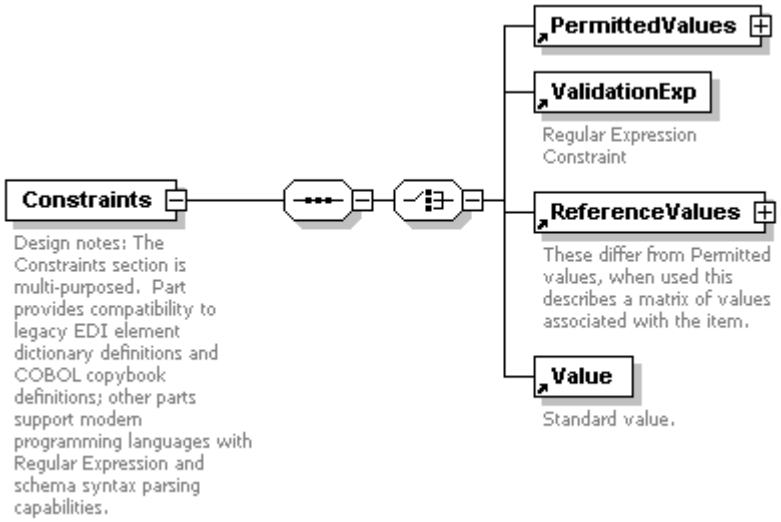
diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	DefaultValue ValueRule ExternalValueList ValueItem
used by	element PhysicalDetail/Constraints

element **PhysicalDetail**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	Constraints

used by	elements	Assembly/BaseDetails Assembly/ExtendedDetails/Associations/Components/Component
annotation	documentation documentation	Describes physical attributes of the component.Design notes: PhysicalDetail is provided here as an option. This allows inline declarations of simple physical components where appropriate, particularly of part child items that will not be used later as standalone entities. Typically however, this will be eschewed in favour of providing a lookup reference value of the related item (usually UID) contained in a separate CRI definition. The PhysicalDetail section of the CRI allows implementation of the full information characteristics of the core component in the real world context. An example is a date core component, that is then physically detailed as Month / Day / Year structural encoding. Each of these individual items is documented further below.

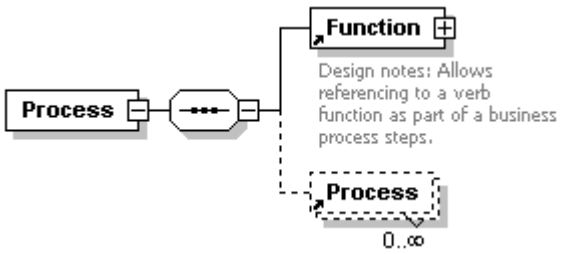
element **PhysicalDetail/Constraints**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	PermittedValues ValidationExp ReferenceValues Value

attributes	Name	Type	Use	Default	Fixed
	minLength	xs:int	optional	1	
	maxLength	xs:int	optional	99	
	baseDataType	baseDataTypes	optional	string	

annotation	documentation	Design notes: The Constraints section is multi-purposed. Part provides compatibility to legacy EDI element dictionary definitions and COBOL copybook definitions; other parts support modern programming languages with Regular Expression and schema syntax parsing capabilities.

element **Process**

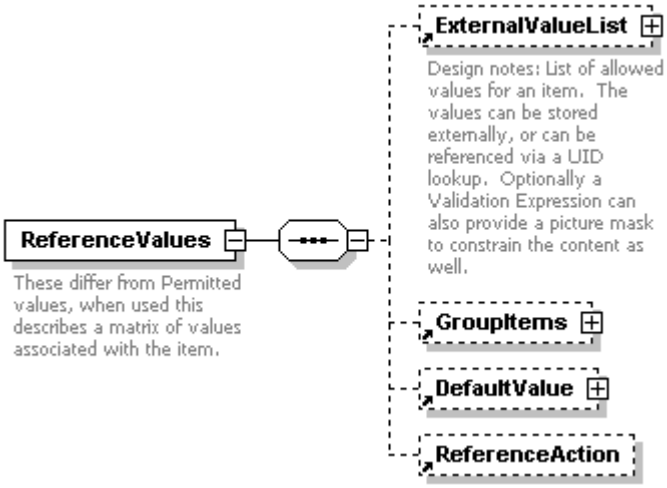
diagram	
namespace	http://www.unece.org/cefact/cri/1.0

children	Function Process				
used by	elements Process Assembly/ExtendedDetails/Processes				
attributes	Name	Type	Use	Default	Fixed
	name	xs:string	required		
	classification	xs:string	optional		
	UIDReference	xs:string	optional		

element **ReferenceAction**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
used by	element ReferenceValues				
attributes	Name	Type	Use	Default	Fixed
	functionName	xs:string	required		
	reference	xs:string	required		
	taxonomy	xs:NMTOKEN	required		
	registry	xs:string			

element **ReferenceValues**

diagram		
namespace	http://www.unece.org/cefact/cri/1.0	
children	ExternalValueList GroupItems DefaultValue ReferenceAction	
used by	elements PhysicalDetail/Constraints Function	
annotation	documentation documentation	These differ from Permitted values, when used this describes a matrix of values associated with the item.Since the structure of these values is unknown, a function is associated to process them.

element **SetValue**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0

used by	elements **DefaultValue MaxValue MinValue**				
attributes	Name	Type	Use	Default	Fixed
	assignedValue	xs:string			
	computeValueRuleUID	xs:string			
	action	xs:NMTOKEN	optional	assign	

686
687
688 element **ValidationExp**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
used by	elements **PhysicalDetail/Constraints ExternalValueList**				
attributes	Name	Type	Use	Default	Fixed
	mask	xs:string	required		
	type	regexpTypes	required		
	comment	xs:string	optional		
annotation	documentation Regular Expression Constraint				

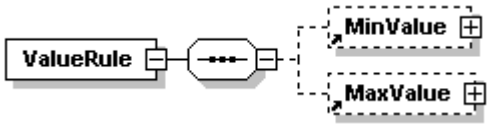
689
690
691 element **Value**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
used by	element **PhysicalDetail/Constraints**				
attributes	Name	Type	Use	Default	Fixed
	default	xs:boolean	optional		
annotation	documentation Standard value.				

692
693
694 element **ValueItem**

diagram					
namespace	http://www.unece.org/cefact/cri/1.0				
used by	elements **GroupItems PermittedValues**				
attributes	Name	Type	Use	Default	Fixed
	displaycode	xs:string			
	value	xs:string	required		
	comment	xs:string			

element **ValueRule**

diagram	
namespace	http://www.unece.org/cefact/cri/1.0
children	**MinValue** **MaxValue**
used by	element **PermittedValues**

simpleType **baseDataTypes**

namespace	http://www.unece.org/cefact/cri/1.0
type	restriction of **xs:string**
used by	attribute **PhysicalDetail/Constraints/@baseDataType**
facets	enumeration time enumeration datetime enumeration string enumeration double enumeration float enumeration number enumeration integer enumeration text enumeration boolean enumeration list enumeration char enumeration byte enumeration codevalue enumeration currency
annotation	documentation Possible values for PhysicalDetail@baseDataType

simpleType **contextTypes**

namespace	http://www.unece.org/cefact/cri/1.0
type	restriction of **xs:string**
used by	attribute **Context/Rule/@type**
facets	enumeration GeoPolitical enumeration BusinessProcess enumeration Industry enumeration Language enumeration Product enumeration Platform enumeration Custom
annotation	documentation Possible values for Context@type.

simpleType **locales**

namespace	http://www.unece.org/cefact/cri/1.0
type	restriction of **xs:string**

used by	attributes **CoreComponent/@defaultLocale** **Assembly/@locale** **Context/@locale**
facets	enumeration da_DK enumeration de_AT enumeration de_AT_EURO enumeration de_CH enumeration de_DE enumeration de_DE_EURO enumeration de_LU enumeration de_LU_EURO enumeration el_GR enumeration en_CA enumeration en_GB enumeration en_IE enumeration en_IE_EURO enumeration en_US enumeration es_ES enumeration es_ES_EURO enumeration fi_FI enumeration fi_FI_EURO enumeration fr_BE enumeration fr_BE_EURO enumeration fr_CA enumeration fr_CH enumeration fr_FR enumeration fr_FR_EURO enumeration fr_LU enumeration fr_LU_EURO enumeration it_CH enumeration it_IT enumeration it_IT_EURO enumeration ja_JP enumeration ko_KR enumeration nl_BE enumeration nl_BE_EURO enumeration nl_NL enumeration nl_NL_EURO enumeration no_NO enumeration no_NO_B enumeration pt_PT enumeration pt_PT_EURO enumeration sv_SE enumeration tr_TR enumeration zh_CN enumeration zh_TW enumeration OTHER
annotation	documentation List of all locales supported.

simpleType **regexpTypes**

namespace	http://www.unece.org/cefact/cri/1.0
type	restriction of **xs:string**
used by	attribute **ValidationExp/@type**
facets	enumeration POSIX enumeration GNU enumeration PERL enumeration JAKARTA enumeration SQL enumeration COBOL enumeration EDI enumeration OTHER
annotation	documentation Possible values for Regexp@type

A Addendum

Sample Core Component XML instances. These have been converted directly from the latest core component library specifications (Word documents) using an automated process that extracts to a text file delimited format and hence to the CRI format. These are of course logical CRI entries, not physical CRI entries.

The full details are available as two separate XML instance files; only a fragment is given here as illustrative documentation.

Figure 7. Fragment of Core Components

```
<?xml version="1.0" encoding="UTF-8" ?>
 <CoreComponents>
  <CoreComponent defaultLocale="en_US" type="Noun" model="logical">
   <MetaInformation>
     <annotation>
        <documentation type="description">UN default code
          component</documentation>
       </annotation>
     <Identifiers>
        <Identifier type="UID" value="UN000105" />
       </Identifiers>
       <Behaviour isRestrictable="true" isExtensable="true" />
     <ExplanationOfUse>
        <Usage type="assemblydoc" instance="element" />
       </ExplanationOfUse>
    </MetaInformation>
   <DefaultAssembly>
     <Assembly locale="en_US">
       <BaseDetails>
          <ElementName>Amount</ElementName>
          <Label>Amount</Label>
          <Description extendedDescription=""
            extendedMIMEtype="HTML">A number of monetary
            units specified in a currency where the unit of
            currency is explicit or implied.</Description>
          <ExtensionDefinitions />
        <PhysicalDetail>
           <Constraints baseDataType="string" />
         </PhysicalDetail>
       </BaseDetails>
      <ExtendedDetails>
          <Processes />
        <Associations>
```

```
                <Association reference="UN000105"
                  taxonomy="UID" relationship="equivalent"
                  registry="" />
              <Components>
                <Component name="Amount. Content
                  (000106) - Amount Currency.
                  Identification. Code (000107)" use="any"
                  relation="any" direction="either"
                  UIDreference="UN000105" taxonomy="other"
                  registry="" />
              </Components>
            </Associations>
            <Schemas />
            <Context locale="" />
          <Model>
            <ObjectClass>Amount</ObjectClass>
            <PropertyTerm>Type</PropertyTerm>
          </Model>
        </ExtendedDetails>
      </Assembly>
    </DefaultAssembly>
  </CoreComponent>
<CoreComponent>
  <MetaInformation>
    <annotation>
      <documentation type="description">UN default code
        component</documentation>
    </annotation>
    <Identifiers>
      <Identifier type="UID" value="UN000089" />
    </Identifiers>
    <Behaviour isRestrictable="true" isExtensable="true" />
    <ExplanationOfUse>
      <Usage type="assemblydoc" instance="element" />
    </ExplanationOfUse>
  </MetaInformation>
  <DefaultAssembly>
    <Assembly locale="en_US">
      <BaseDetails>
        <ElementName>Code</ElementName>
        <Label>Code</Label>
        <Description extendedDescription=""
          extendedMIMEtype="HTML">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.</Description>
        <ExtensionDefinitions />
      <PhysicalDetail>
```

```
806                 <Constraints baseDataType="string" />
807               </PhysicalDetail>
808             </BaseDetails>
809           <ExtendedDetails>
810               <Processes />
811             <Associations>
812                 <Association reference="UN000089"
813                   taxonomy="UID" relationship="equivalent"
814                   registry="" />
815               <Components>
816                   <Component name="Code. Content (000091) -
817                     Code List. Identifier (000092) - Code List.
818                     Agency. Identifier (000093) - Code List.
819                     Version. Identifier (000099) - Code. Name
820                     (000100) - Language. Code (000075)"
821                     use="any" relation="any" direction="either"
822                     UIDreference="UN000089" taxonomy="other"
823                     registry="" />
824               </Components>
825             </Associations>
826             <Schemas />
827             <Context locale="" />
828           <Model>
829               <ObjectClass>Code</ObjectClass>
830               <PropertyTerm>Type</PropertyTerm>
831             </Model>
832           </ExtendedDetails>
833         </Assembly>
834       </DefaultAssembly>
835     </CoreComponent>
836
837   </CoreComponents>
```