ebXML Concept - Context and Re-Usability of Core Components

ebXML Core Components

March 23, 2001

Version 1.02

1 Status of this Document

This is an ebXML specification for approval by the eBusiness community.

The document formatting is based on the Internet Society's Standard RFC format.
This document has not been harmonized with the ebXML Core Components efforts.
Distribution of this document is limited to the ebXML community.

This version:

Contetx and Re-Usability of Core Components Ver 1.02

2 ebXML participants

We would like to recognize the following for their significant participation to the development of this document.

Editing team:	Mike Adcock, APACS
	Sue Probert, Commerce One
	James Whittle, e CentreUK
	Gait Boxman, TIE
	Thomas Becker, SAP
Contributors:	
	Tom Warner
	Jim Dick
	Rob Jeavons
	David Connelly
	Arofan Gregory
	Martin Bryan
	Mike Adcock
	Eduardo Gutentag
	Matthew Gertner
	Polly Jan
	Sharon Kadlec
	Sally Wang
	James Wertner
	Todd Freter
	Henrik Reiche
	Chris Nelson
	Martin Roberts
	Samantha Rolefes

3 Table of Contents

4 Introduction

4.1 Summary of Contents of Document

This document describes the contextual categories that have been identified as most critical in describing the use of generic Core Components for business information purposes. It also suggests source lists of context such as, for example ISO 3166 for country related contexts.

The document will also describe how new context categorisations can be added and used. This might include adding new categories, or refining existing ones. The refinement may include both addition and subtraction of sections of a context taxonomy.

This document contains the context definitions, the recommended sources; and examples of how these contexts may be applied in business use. This document should be read in conjunction with “Initial Catalogue of Context Drivers ver 1.02” document.

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 RFC 2119 [Bra97].

4.2 Context Defined

When a business process is taking place, the context in which it is taking place can be specified by a set of contextual categories and their associated values. For example, if an auto manufacturer is purchasing paint from a chemical manufacturer, the context values might be as follows:

Contextual Category	**Value**
Process	Procurement
Product Classification	Paint
Region (buyer)	France
Region (seller)	U.S.
Industry (buyer)	Not required (generic)
Industry (seller)	retail

The following set of scenarios explain when context may be applied to a specific Core Component:

- Design Time - to create the minimum useful schema.
- Integration Time - Identify and help resolve data requirements conflicts required for business transactions.

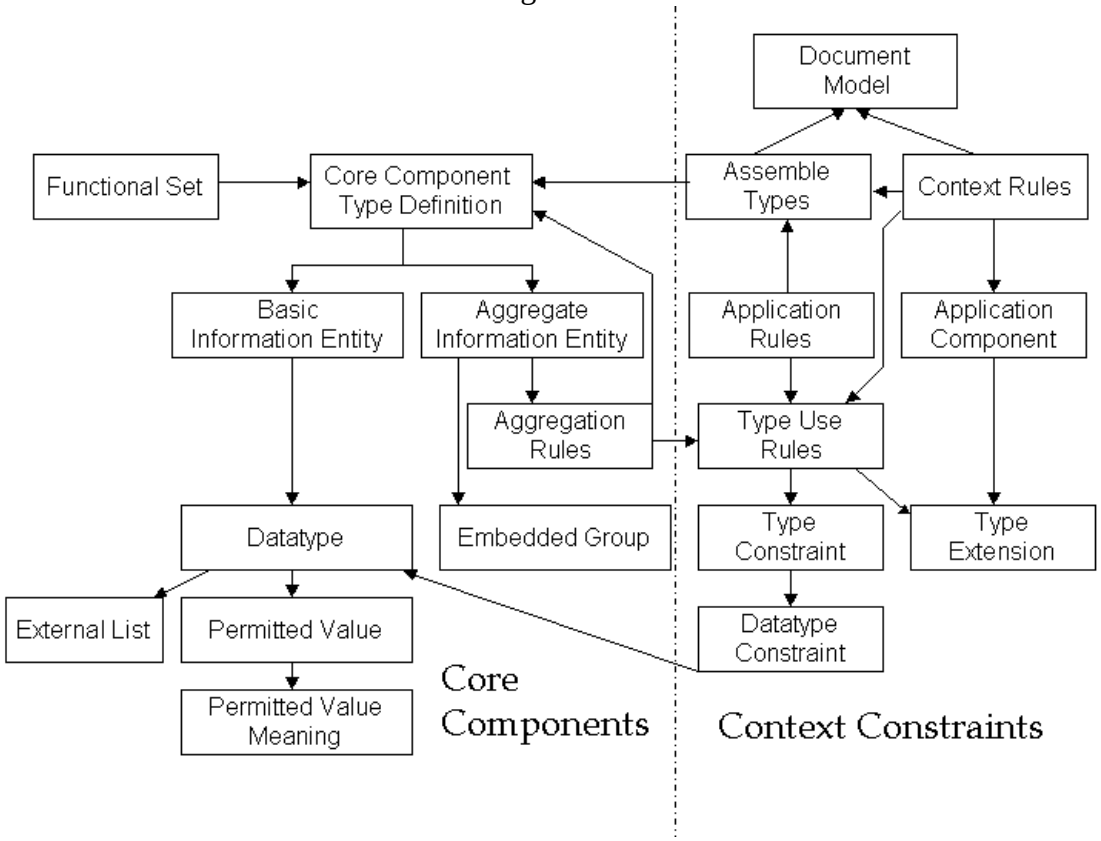
- Run Time - to express the business relationships between data.
  - Used by Trading Partners to validate the runtime document instances.
- Navigation of the registry to find other data sets.
  - Need to hold the data about the context in the rules.
- Discovery Process for creating Core Components or extensions.
  - Core Components are discovered along with the business context in which they are used.

Rules indicate which context values (or combination thereof) must be present in order for them to be applied to a specific core component, as well as the action to be undertaken if a match occurs. Actions include adding additional information to a functional unit, making this information optional, required or eliminating optional information. We might, for instance, specify that addresses associated with organizations in the U.S. region be required to include a state (which might otherwise be optional). Note that these contextual changes are made individually to the Core Components that make up a business document, and not to the business document itself. Core components are always used within a context.

5 Using Context Descriptors

5.1 Context-controlled Core Component Metamodel

The diagram below summarises the formal model for the Context Controlled Core Component Metamodel. The left-hand side of this diagram identifies the units specific to the definition of Core Components. The right-hand side of the diagram identifies the units used in defining Context Constraints. (That mechanism that functions to alter the core components for use within a specific business context.) Type Use Rules are used both to manage component reuse within core components and to manage constraints within Context Constraints. The formal model for the Context-controlled Core Component Metamodel can be seen in "Initial Catalogue of Context Drivers ver 1.02" document.



Despite this underlying simplicity, complications inevitably arise that make context rules impossible to implement outside of a complete specification of business process requirements for a real-world scenario. Broadly speaking, these complications relate to scenarios where two rules both match the context, but have conflicting results, or where different results are reached depending on the order in which matching rules are applied. The following examples illustrate these two cases (also see "Initial Catalogue of Context Drivers ver 1.02).

- One rule could require that if the buyer is in the U.S. region, product description should not be included in invoice line items. Another specifies that if the seller is in France, the product description (in French) shall be included.
- One rule could require that if the buyer's industry is automotive, the product category should be added to the invoice line items. Another specifies that if a product category information entity exists and the seller's industry is chemicals, an attribute should be added to the product category to indicate the toxicity of the products in the category. If the toxicity requirement were applied first, the attribute would not be added (since the product category was not yet present). The outcome therefore depends on the order in which the rules are applied.

The problem with these types of situations is not so much that there is no way to resolve them. It is rather that there are many possible solutions with no clear way of deciding which to choose, and all are sufficiently complex to place a significant burden on the implementer.

Additional complications result from the potentially hierarchical nature of context values. For example, the possible values for region belong in a hierarchical space (e.g. continent, country, region, city, etc.). The region specification can therefore be very general or very specific. Since rules can match a general value (e.g. apply if the organization is in North America) or a specific value (e.g. apply if the organization is in Omaha, Nebraska), there must be some way of determining which rules to apply (any combination including all of them) if several match. This is because, in some cases, a specific rule may complement the general rule, while in others it may override it.

5.1.1 Core Component Type Definitions

A Core Component Type Definition defines a reusable type of core component for which no pre-determined use name has been assigned." to "A Core Component Type Definition defines a reusable type of core component, to which a pre-determined use name has not been assigned.

Each definition is given a globally unique Identifier, which should be suitable for use as a registry or registry key.

A human-readable name for the type (ending in the word Type, e.g. AddressType), and a brief description of the purpose of the type, are also required. For further specification see the "ebXML CC Dictionary Entry Naming Conventions Ver 1.02" document.

By default a Core Component Type Definition is deemed to be restrictable or extendable. If this is not the case the isRestrictable or isExtendable boolean properties must be set to False.

5.1.2 Basic Information Entity

Where the types of data that are permitted for a Basic Information Entity are defined by an external agency the name of the maintaining agency and the agency assigned identifier (id) must be recorded.

A formal definition of the relevant Datatype, defined in accordance with Part 2 of the XML Schema specification, must be associated with each Basic Information Entity.

If a data type is associated with an externally defined list of permitted values, then the URI of a resource that defines the set of currently approved permitted values should be recorded as an external value list object.

If the list of permitted values is defined as part of the core component definition a Permitted Value List must be created. The list consists of one or more Permitted Values identified by a name that is unique within the list, each of which should be assigned one or more Permitted Value Meanings, each of which consists of a statement of the meaning assigned to the value and the IETF RFC1766 language code identifying the language in which the meaning has been defined.

5.1.3 Aggregate Information Entity

For each component forming part of an Aggregate Information Entity an Aggregation Rules that identifies a Type Use Rules object must be created. The Type Use Rules record the Name assigned to the referenced type within the location and, optionally, an explanation of the use to which the embedded component is being put within this component.

Where there are constraints on the number of times an embedded component can be used these are recorded as the MinMaxConstraints property.

Where there are constraints on the order in which sub-components within the aggregate are to be used an Embedded Group must be defined to identify whether the constraint applies to the use of a choice or sequence of objects.

5.1.4 Functional Set

A Functional Set is a set of two or more Core Component Type Definitions or Functional Sets that can be used to model information related to a single function in different ways.[1]

[1] For example, a location could be recorded as a postal address, a United Nations location code or as a set of co-ordinates as generated by a Global Positioning System. Which of this set of equivalent functions would be chosen for a particular message is context dependent.

5.2 Context Constraints

A Document Model is created by applying a set of Context Rules to a set of Core Component Type Definitions that have been “assembled” to meet a defined business process.

The Assemble Types modelling element identifies the base Core Component Type Definitions, applies an appropriate sequence to the components and renames embedded components as required within the business process.

The Context Constraints define modifications to be made to existing Core Component Type Definitions when used within specific contexts, and any Application Component needed to extend a core component or the document model.

Individual constraints are associated with a particular value within a named taxonomy stored as a named context classification within an ebXML repository.

Where the constraint requires that the base definition of a core component be redefined the constraints are defined as a Type Constraint. Where the constraint applies to a facet of a Datatype definition it forms a Datatype Constraint that is associated with a specific Datatype.

5.3 Seeding Core Components

Lower level core components, either basic or aggregate information entities, can be re-used within higher level aggregates. Fundamentally, they are used "in the context of" the higher level aggregate. This is a purely structural context, not a business context, creating stereotype (i.e. fundamental or generic) information entities.

Recognizing that there are situations in which equivalent information can be expressed in several ways, relevant core components can be grouped together into Functional Sets. These provide a means by which a limited choice of stereotype information entities can be offered as alternative ways of specifying information for a particular function, e.g. a location can be specified as an address, a GPS reference, or a UN Locode. While the functional set is still a stereotype, the choice is dependent on a business context or contexts.

5.4 Using Core Components

Use of a core component without any modification in a particular business context creates a Substitute Information Entity. This is registered under a unique business name formed from the context and the stereotype component names.

Note: This is essential to record the industry sector(s) that use the substitute information entity, the context(s) in which they are used, and all the substitute information entities that use the Core Component.

Use of a core component with extensions (or indeed restrictions) in a particular business context creates a Process Specific Entity. This is registered under a unique business name formed from the context and the stereotype component names.
Note: This is essential to record the industry sector(s) that use the substitute information entity, the context(s) in which they are used, and all the process specific entities that use the Core Component.

Substitute information entities and process specific entities are collectively Context Constrained Information Entities. Registration of all these, however numerous, is essential to achieve maximum re-use, to avoid "re-inventing the wheel", and to gain interoperability.

5.5 Building Business Documents

Business documents are built by drawing on the repository "library" of components. The context descriptors that are registered for each component are used to select the appropriate context constrained information entities for the business document that is being built. These values would be the same as values found in a business process model that informs the contextual use of the core components.

If no appropriate context constrained information entity exists, a new one must be created, according to the principles described in the previous section, and ideally using an existing stereotype. Registration of the new process specific information entity adds to the range of available context descriptors.

5.6 Beyond Re-use

If no appropriate existing stereotype exists, an industry grouping may need to:

- create additional Basic components for pieces of information which cannot be represented using already-defined Core Components. These are Domain Basic Components.
- use Core Component(s) to construct a non-core Aggregate Component, called a Domain Complex Component.
- use Core Component(s) and Domain Components to construct a non-core Complex Component, also known as a Domain Complex Component.
- use Domain Component(s) to construct a non-core Complex Component. These are also Domain Complex Components.

Ideally, Domain Components need to be recorded in the same detail as Core Components, complete with relevant Context(s). They are part of extensibility and should be registered so as to avoid 're-inventing the wheel'. Newcomers can re-use Domain Components and register any additional Context(s).

At some point, non-core Domain Components can become Core Components, according to criteria that judge the degree of re-use. These values would be the same as values

found in a business process model that informs the contextual use of the core components.

5.7 Non-compliance Issue

This section raises two basic issues:

1) Extensibility
2) Registration

Registering Domain Components cannot be completely policed. Groups or companies might decide to use Core Components, extend them and invent their own Domain Components and never register them.

As a consequence, the use of these Domain Components will not become part of the ebXML standards community. Exact equivalents may well be re-invented in a different way, with different naming, and formally registered as a Domain Components.

Unregistered Domain Components:

- will hinder communication and interoperability between different communities.
- should not, in any circumstances, be favored over formally registered equivalents.

6 Disclaimer

INCLUDEPICTUREThe 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 design.

7 Contact Information

Team Leader

Name	Arofan Gregory
Company	Commerce One
Street	Vallco Parkway
city, state, zip/other	Cupertino, CA
Nation	US
Phone:	
EMail:	arofan.gregory@commerceone.com

Vice Team Lead

Name	Mike Adcock
Company	APACS
Street	Mercury House, Triton Court, 14 Finsbury Square
city, state, zip/other	London EC2A 1LQ
Nation	UK
Phone:	+44-20-7711-6318
EMail:	mike.adcock@apacs.org.uk

Editor

Name	James Whittle
Company	e centre[UK]
Street	10, Maltravers Street
city, state, zip/other	London WC2R 3BX
Nation	UK
Phone:	+44-20-7655-9022
EMail:	james.whittle@e-centre.org.uk

8 Copyright Statement

To be defined