Document Assembly and Context Rules

ebXML Core Components

10 May 2001
Version 1.04

1   Status of this Document

14

15 This Technical Report document has been approved by the Core Component Project
16 Team and has been accepted by the ebXML Plenary.
17
18 This document contains information to guide in the interpretation or implementation of
19 ebXML concepts.
20
21 Distribution of this document is unlimited.
22
23 The document formatting is based on the Internet Society's Standard RFC format.
24
25 This version:
26   www.ebxml.org/specs/ebCCDOC.pdf
27
28 Latest version:
29   www.ebxml.org/specs/ebCCDOC.pdf
30
31

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
Team Leader:	
	Arofan Gregory, Commerce One
Vice Team Leader:	
	Eduardo Gutentag, SUN Microsystems
Contributors:	
	Eduardo Gutentag
	Arofan Gregory
	Matthew Gertner
	Martin Bryan
	Martin Roberts
	Lauren Wood
	Chris Nelson
	Todd Freter
	Mike Adcock

59 # 3 Table of Contents

86

87 **4 Introduction**

88 The challenge of ebXML is to create a framework for automating trading partner
89 interactions that is both:
90 • Sufficiently generic to permit implementation across the entire range of business
91   processes (in various industries, geographical regions, legislative environments, etc.)
92 • Expressive enough to be more effective than ad hoc implementations between
93   specific trading partners.
94
95 This specification document describes the way in which rules can be formed and/or
96 derived, but is not a prescriptive specification. It is believed that rule mechanisms will be
97 achieved in different ways within different implementations/solutions.
98
99 This document deals with two specific aspects of the task:
100 • The assembly of core component schemas into full business document schemas,
101 • The modelling of core components for business documents that provide useful
102   building blocks for real-world trading scenarios and, at the same time, are open
103   enough to take into account the wide variety of document formats required by
104   organizations with differing business practices and requirements.
105
106 Complicating this situation is the need for interoperability: companies must be able to
107 communicate business documents effectively with minimum human intervention, even
108 though the formats used may have a significantly different syntax.
109
110 Central to achieving this goal is the notion of context. Context provides a framework for
111 adapting generic core components to specific business needs, while keeping the
112 transformation process transparent so that the processing engine can find a useful set of
113 common information for use by different trading partners. An example of a contextual
114 category that is useful for business is industry: different industries will have different
115 requirements for the syntax of core components. By starting with a generic core
116 component and using context to derive a context-specific core component, we ensure
117 that, at the very least, the information in the generic component will be useful when
118 interacting with a trading partner in a different context (i.e. industry, region, etc.). This
119 should be contrasted with the alternative: context-specific business documents that are
120 not built from generic core components and therefore provide no common basis for
121 interaction outside of that context.
122
123 In order to assemble full business documents from core components, rules are drawn
124 specifying what components are to be included in the document, and how.
125
126 In order to generate a context-specific core component, rules are associated with different
127 values for each of the context categories. This document presents a proposed syntax for
128 these context rules, and a methodology for applying them, in order to achieve maximum
129 reuse of existing XML software development tools and libraries.
130

131   The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD,
132   SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this
133   document, are to be interpreted as described in RFC 2119.

4.1 Summary of Contents of Document

135   This specification describes the mechanism for assembling documents from the library of
136   Core Components. It describes the process of refining the components to contain exactly
137   the information required by a specific business context and describes the output of this
138   process such that it enables interoperability independent of any syntax binding. This
139   approach also lends itself to an automated comparison with other, similar document
140   definitions created in other syntaxes. The provided specifications are;
141   • A syntax for providing the assembly rules, with a DTD and sample;
142   • A syntax for refining the assembled structures, and indicating specific context
143   drivers, also with DTD and sample;
144   • A format for capturing the critical information about the final result, provided as
145   an XML DTD.

4.2 Related Documents

147   As mentioned above, other documents provide detailed definitions of some of the
148   components and their inter-relationship. They include ebXML Specifications on the
149   following topics:
150   • [ebCNTXT] Context and Re-Usability of Core Components Ver 1.04
151   • [ebCCNAM] Naming Convention for Core Components Ver 1.04
152   • [ccOVER] Catalogue of Core Components Ver 1.04

5 Document Assembly

154 Document assembly is the rules-based process whereby Core Components are extracted
155 from the repository and used to create a schema model. That can then be used to create an
156 XML schema which, when appropriate, and after the application of any relevant Context
157 Rules, can be used to validate the contents of a business document.
158
159 For example, a Purchase order schema may consist of two parties (buyer, seller), and a
160 sequence of items. Purchase orders are not Core Components; they must be assembled
161 out of Core Components found in the repository.

162 **6 Context Rules**

163 When a business process is taking place, the context can be specified by a set of
164 contextual categories and their associated values. For example, if an auto manufacturer is
165 purchasing paint from a chemicals manufacturer, the context values might be as follows:
166

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

167
168 Rules indicate which context values (or combination thereof) must be present in order for
169 them to be applied, as well as the action to be undertaken if a match occurs. Actions
170 include adding additional information to a functional unit, making this information
171 optional, required or eliminating optional information. We might, for instance, specify
172 that addresses associated with organizations in the U.S. region be required to include a
173 state (which might otherwise be optional). Note that these contextual changes are made
174 individually to the Core Components that make up a business document, and not to the
175 business document itself.
176
177 Despite this underlying simplicity, complications arise in certain cases that make real-
178 world implementation of context rules extremely tricky. Broadly speaking, these
179 complications relate to scenarios where two rules both match the context, but have
180 conflicting results, or where different results are reached depending on the order in which
181 matching rules are applied. The following examples illustrate these two cases (and refer
182 to the sample context given above; see also see the document ebXML TR - Catalogue of
183 Context Drivers Ver 1.04):
184   • One rule could require that if the buyer is in the U.S. region, product description
185   should not be included in invoice line items. Another specifies that if the seller is
186   in France, the product description (in French) shall be included.
187   • One rule could require that if the buyer's industry is automotive, the product
188   category should be added to the invoice line items. Another specifies that if a
189   product category information entity exists and the seller's industry is chemicals,
190   an attribute should be added to the product category to indicate the toxicity of the
191   products in the category. If the toxicity requirement were applied first, the
192   attribute would not be added (since the product category was not yet present). The
193   outcome therefore depends on the order in which the rules are applied.
194
195 The problem with these types of situations is not so much that there is no way to resolve
196 them. It is rather that there are many possible solutions with no clear way of deciding

197   which to choose, and all are sufficiently complex to place a significant burden on the
198   implementer.
199
200   Additional complications result from the potentially hierarchical nature of context values.
201   For example, the possible values for region belong in a hierarchical space (e.g. continent,
202   country, region, city, etc.). The region specification can therefore be very general or very
203   specific. Since rules can match a general value (e.g. apply if the organization is in North
204   America) or a specific value (e.g. apply if the organization is in Omaha, Nebraska), there
205   must be some way of determining which rules to apply (any combination including all of
206   them) if several match. This is because, in some cases, a specific rule may complement
207   the general rule, while in others it may override it.

208 # 7 XML-Based Rules Model

209 The custom XML syntax for assembly and context rules presented in this document is
210 designed to ensure an appropriate level of abstraction for the rules, and to allow them to
211 be applied both manually and/or by programs.

212 ## *7.1 Rules Syntax*

213 The syntax is presented here in tabular form, to avoid tying the definition of the schemas
214 it describes to a given schema language syntax. This table should be sufficiently
215 expressive to permit the derivation of a corresponding schema definition in various
216 concrete schema syntaxes (DTD, XML Schema, SOX, XDR, etc.). This syntax describes
217 two XML schemas describing two classes of XML documents whose roots are,
218 respectively, `<Assembly>` and `<ContextRules>`. They are presented here in a
219 single table because there is conceptual commonality.
220
221 A specific rules file is thus an XML document conforming to one of these schemas.
222
223 The following values are allowed for the occurrence field:
224

Name	Meaning
Required	Must occur exactly once
Optional	May occur once at most
+	Required and may occur multiply
*	Optional and may occur multiply
(m,n)	Occurs at least m and at most n times

225
226 Names separated by the vertical bar (|) represent a disjunction (i.e. one and only one of
227 the list of names may occur). For example, Apple|Orange|Banana indicates that either an
228 Apple or an Orange or a Banana may occur in this location.
229 Names prefixed with the commercial at sign (@) are represented as attributes in the XML
230 instance (and the leading @ is removed from the attribute name).
231

Name	Type	Occurrence	Default	Description
Assembly				
Assemble	complex	+		List of assembled Core Components
@name	string	optional		Name of collection of assembled document schemas.
@version	string	optional		Version of the Assembly Rules document.
Assemble				
CreateElement	complex	+		List of Core Components

CreateGroup	complex	*		Create a group of elements
@name	string	required		Name of the document schema being assembled
CreateGroup				
@type	enum	default	sequence	Type of group to be created (the only permitted values are 'sequence' and 'choice')
CreateGroup	complex	*		Create a group of elements
CreateElement	complex	*		Create an Element
UseElement	complex	*		Use the named element from among the children of the element being created.
Annotation	complex	*		Insert Annotation
CreateElement				
Type	string	optional		Type of element to be created
MinOccurs	string	optional		Minimum occurrences for the element created
MaxOccurs	string	optional		Maximum occurrences for the element created. One possible value (other than integer) is 'unbounded'.
@id	ID	required		Id of the created element
@idref	IDREF	optional		Reference to the ID of another created element
Name	string	required		Name of the element to be assembled
@location	UUID|URI	required		Location of the element to be assembled (i.e. query to the registry)
Rename	EMPTY	optional		Renames children of the created element
Annotation	complex	*		Insert Annotation
Rename				
@from	string	required		Original name of the child element being renamed
@to	string	required		New name of the child being renamed
ContextRules				
Rule	complex	+		List of rules to be applied
@version	string	optional		Version of the ContextRules document.

Rule				
@apply	enum	default	exact	(See below)
Condition	complex	required		When rule should be run
Action	complex	+		What happens when rule is run
@order	integer	default	0	Defines order for running rules. Rules with higher value for order are run first
Taxonomy	EMPTY	+		List of taxonomies used in a Rule that employs hierarchical conditions.
Taxonomy				
@ref	URI	Required		Pointer to a taxonomy.
Condition				
@Test	string	Required		Boolean expression testing whether the rule should be run. Uses XPath syntax [XPATH]
Action				
@applyTo	string	Required		Node to apply action to
Add|Subtract|Occurs	complex	+		List of modifications to content model
Add				
MinOccurs	integer	default	1	Minimum number of times that the new instance must occur
MaxOccurs	integer	default	1	Maximum number of times that the new instance can occur
@before	string	optional		Specifies before which child the addition should occur.
@after	string	optional		Specifies after which child the subtraction should occur.
Element	complex	optional		Adds a new element to the content model.
Attribute	complex	optional		Adds a new attribute to the content model
Annotation	complex	*		Insert Annotation
Subtract				
Element	complex	optional		Removes an element from the content model.
Attribute	Complex	optional		Removes an attribute from the content model
Occurs				

Element	complex	required		Changes an optional element to required.
MinOccurs	integer	optional	1	Overrides the minimum number of occurrences for this Element.
MaxOccurs	integer	optional	1	Overrides the maximum number of occurrences for this Element.
Element				
Nameame	string	required		Name of element to be modified
Type	string	optional		Type of element, required only if contained in an Add tag
Attribute	complex	*		Attribute(s) of this element
Annotation	complex	*		Insert Annotation
Attribute				
Name	string	optional		Name of attribute to be modified
Type	string	optional		Type of the attribute (e.g. ID, CDATA, enumerated list, etc.)
Use	required | optional | fixed | default	optional	required	Indicates whether required or optional, and if the latter whether fixed or defaulted
Value	string	optional		Indicates a fixed or defaulted value, or a value to be modified
UseElement				
Name	string	required		Name of the element being used
Annotation	complex	*		Insert Annotation
Comment				
	string	optional		Ubiquitous. Records comments about the rules document at the location it appears. It is not intended to be output in the result document.
Type				
	string	optional		Type in the output
MinOccurs				
	string	Optionl		Minimum number of occurrences in the output
MaxOccurs				

	string	Optional		Maximum number of occurrences in the output

7.1.1   Notes on Assembly

232
233 The MinOccurs and MaxOccurs elements in the CreateElement element specify the
234 occurrence indicator that the created element will have in the resulting schema. Thus, an
235 element created with `<MinOccurs>1</MinOccurs>` and
236 `<MaxOccurs>1</MaxOccurs>` should be specified in the resulting schema as an
237 element that must occur only once.
238
239 An `<Assembly>` may contain more than one assembled document schema. Whether a
240 separate document is output for each assembled schema is implementation dependent.
241

7.1.2   Notes on Context

242
243 Several built-in variables are used to access context information. These variables
244 correspond to the various context drivers identified in the document ebXML TR -
245 Catalogue of Context Drivers Ver 1.04:
246   • Industry
247   • Business Process
248   • Product
249   • Geopolitical
250   • Official Constraints
251   • Role
252 All of these variables have values of type string.
253
254 The "Apply" attribute of the "Rule" element type is used for determining the behaviour
255 of rules that use hierarchical value spaces. Possible values are "exact" (match only if the
256 value in the provided context is precisely the same as that specified in the rule) and
257 "hierarchical" (match if the value provided is the same or a child of that specified in the
258 rule). For example, if the rule specifies the region "Europe", the value "France" would
259 match only if the "Apply" attribute is set to "hierarchical" ("exact" being the default).
260
261 The minOccurs and maxOccurs elements in Occurs are defaulted. If neither is present,
262 the intent is to change an optional element into a required one (that is, it's a shortcut for
263 `<MinOccurs>1</MinOccurs>, <MaxOccurs>1</MaxOccurs>`).
264
265 The <Attribute> element has four optional elements in its content model, of which at
266 least one must be present. If the value of the applyTo attribute of Action is an attribute,
267 there is no need to specify the Name again. If only Value is specified, the intention must
268 be to add or subtract a given value from an attribute's enumerated list.
269
270 Rules apply only to the source. For instance, given a source that contains an optional
271 element type named 'X', a rule can be applied to rename 'X' to 'Y', but a rule to make
272 'Y' required, rather than 'X', would be illegal.
273
274 (also see [ebCNTXT] Context & Re-Usability of Core Components Ver1.04)

275 ## *7.2 DTD for Assembly Documents*

```
276  <!ELEMENT Assembly (Assemble+)>
277  <!ATTLIST Assembly
278          version  CDATA #IMPLIED
279          id       ID    #IMPLIED
280          idref    IDREF #IMPLIED
281  >
282
283  <!ELEMENT Assemble (CreateElement|CreateGroup)+>
284  <!-- the name is the name of the schema that is created -->
285  <!ATTLIST Assemble
286          name     CDATA #REQUIRED
287          id       ID    #IMPLIED
288          idref    IDREF #IMPLIED
289  >
290
291  <!ELEMENT CreateGroup
292          (CreateGroup|CreateElement|UseElement|Annotation)+ >
293  <!ATTLIST CreateGroup
294          type  (sequence|choice) "sequence"
295          id       ID    #IMPLIED
296          idref    IDREF #IMPLIED
297  >
298
299  <!ELEMENT CreateElement (Name?, Type?, MinOccurs?, MaxOccurs?,
300  (CreateGroup|Rename|UseElement|Condition|Annotation)*)>
301  <!-- you need either a Name sub-element and
302  an ID attribute, or just an IDREF attribute -->
303  <!-- max can be an integer or the word "unbounded" -->
304  <!ATTLIST CreateElement
305          id       ID    #IMPLIED
306          idref    IDREF #IMPLIED
307          location CDATA #IMPLIED
308  >
309
310  <!ELEMENT Name     (#PCDATA)>
311  <!ELEMENT Type     (#PCDATA)>
312  <!ELEMENT MinOccurs (#PCDATA)>
313  <!ELEMENT MaxOccurs (#PCDATA)>
314
315  <!ELEMENT Rename   EMPTY>
316  <!ATTLIST Rename
317          from     CDATA #REQUIRED
318          to       CDATA #REQUIRED
319          id       ID    #IMPLIED
320          idref    IDREF #IMPLIED
321  >
322
323  <!ELEMENT UseElement  (Annotation|CreateGroup|UseElement)*>
324  <!ATTLIST UseElement
325          name     CDATA #REQUIRED
326          id       ID    #IMPLIED
327          idref    IDREF #IMPLIED
328  >
329
330  <!ELEMENT Condition (Rename|CreateGroup|UseElement|CreateElement)+>
```

```
331   <!ATTLIST Condition
332           test     CDATA #REQUIRED
333           id       ID    #IMPLIED
334           idref    IDREF #IMPLIED
335   >
336   <!ELEMENT Annotation  (Documentation | AppInfo)*>
337   <!ATTLIST Annotation
338           id       ID    #IMPLIED
339           idref    IDREF #IMPLIED
340   >
341
342   <!ELEMENT Documentation  (#PCDATA)>
343   <!ATTLIST Documentation
344           id       ID    #IMPLIED
345           idref    IDREF #IMPLIED
346   >
347
348   <!ELEMENT AppInfo  (#PCDATA)>
349   <!ATTLIST AppInfo
350           id       ID    #IMPLIED
351           idref    IDREF #IMPLIED
352   >
353
```

354 *7.3 DTD for Context Rules Documents*

```
355   <!ELEMENT ContextRules (Rule+)>
356   <!ATTLIST ContextRules
357           version  CDATA #IMPLIED
358           id       ID    #IMPLIED
359           idref    IDREF #IMPLIED
360   >
361
362   <!ELEMENT Rule    (Taxonomy+, Condition+)>
363   <!ATTLIST Rule
364           apply    (exact|hierarchical) "exact"
365           order    CDATA #IMPLIED
366           id       ID    #IMPLIED
367           idref    IDREF #IMPLIED
368   >
369
370   <!ELEMENT Taxonomy    EMPTY>
371   <!-- this ref should be a URI -->
372   <!ATTLIST Taxonomy
373           context  CDATA #REQUIRED
374           ref      CDATA #REQUIRED
375           id       ID    #IMPLIED
376           idref    IDREF #IMPLIED
377   >
378
379
380   <!ELEMENT Condition  (Action|Condition|Occurs)+>
381   <!ATTLIST Condition
382           test     CDATA #REQUIRED
383           id       ID    #IMPLIED
384           idref    IDREF #IMPLIED
```

```
385    >
386
387    <!ELEMENT Action  (Add|Occurs|Subtract|Condition|Comment|Rename)+>
388    <!ATTLIST Action
389            applyTo  CDATA #REQUIRED
390            id       ID    #IMPLIED
391            idref    IDREF #IMPLIED
392    >
393
394    <!ELEMENT Add        ((MinOccurs?,MaxOccurs?,(Element?
395                           |Attribute?))|CreateGroup|Annotation)+>
396    <!-- before and after refer either to the ID of the other element or
397    to its Xpath -->
398    <!ATTLIST Add
399            before   CDATA #IMPLIED
400            after    CDATA #IMPLIED
401            id       ID    #IMPLIED
402            idref    IDREF #IMPLIED
403    >
404
405    <!ELEMENT Rename   EMPTY>
406    <!ATTLIST Rename
407            from     CDATA #REQUIRED
408            to       CDATA #REQUIRED
409            id       ID    #IMPLIED
410            idref    IDREF #IMPLIED
411    >
412
413    <!ELEMENT CreateGroup  (Element)+>
414    <!ATTLIST CreateGroup
415            type  (choice|sequence) "sequence"
416            id       ID    #IMPLIED
417            idref    IDREF #IMPLIED
418    >
419
420    <!ELEMENT Element      (Name, Type?, (Attribute)*, (Annotation)*)>
421    <!ATTLIST Element
422            id       ID    #IMPLIED
423            idref    IDREF #IMPLIED
424    >
425
426    <!ELEMENT Attribute      (Name?, Type?, Use?,
427                              Value?, (Annotation)*)>
428    <!ATTLIST Attribute
429            id       ID    #IMPLIED
430            idref    IDREF #IMPLIED
431    >
432
433    <!ELEMENT Use (#PCDATA)>
434    <!ELEMENT Value (#PCDATA)>
435
436    <!ELEMENT Annotation  (Documentation | AppInfo)*>
437    <!ATTLIST Annotation
438            id       ID    #IMPLIED
439            idref    IDREF #IMPLIED
440    >
```

```
441
442    <!ELEMENT Documentation  (#PCDATA)>
443    <!ATTLIST Documentation
444            id       ID    #IMPLIED
445            idref    IDREF #IMPLIED
446    >
447    <!ELEMENT AppInfo  (#PCDATA)>
448    <!ATTLIST AppInfo
449            id       ID    #IMPLIED
450            idref    IDREF #IMPLIED
451    >
452
453    <!ELEMENT Occurs (MinOccurs?, MaxOccurs?,(Element+))>
454    <!ATTLIST Occurs
455            id       ID    #IMPLIED
456            idref    IDREF #IMPLIED
457    >
458
459    <!ELEMENT Subtract (Element | Attribute)+>
460    <!ATTLIST Subtract
461            id       ID    #IMPLIED
462            idref    IDREF #IMPLIED
463    >
464
465    <!ELEMENT Name    (#PCDATA)>
466    <!ELEMENT Type    (#PCDATA)>
```

468 # 8 Rule Ordering

469 There are two mechanisms for determining the order in which context rules should be
470 applied. The first is document order, that is, the order in which the rules appear in the
471 Rules document. The second is an explicit "Order" attribute that can be used to force a
472 given order on a set of rules. It's an error for two rules have the same order. Users should
473 be careful not to issue rules in an order that would preclude their execution (for instance,
474 adding an attribute to an element that has not been added yet by the rules). Applications
475 must issue error messages when such a situation is encountered, rather than silently
476 ignoring it.

477 # 9   Semantic Interoperability Document

478 This section specifies an XML document format, the Semantic Interoperability Document
479 that a processor applying assembly rules and context rules within a single context can
480 output. This serves two purposes:

481 • It creates a syntax-neutral output format, so that two processors working with
482 different syntax mappings could determine the semantic equivalence of their
483 context rules by comparing the output when expressed in this form.
484 • It provides a mechanism for mapping from a syntax-specific output back to the
485 syntax-neutral one, using techniques such as UUID pointers or Xpath expressions,
486 enabling implementation using existing tools.

487 ## 9.1 DTD for Semantic Interoperability Document

488 The semantic interoperability document type is expressed in the following DTD:
489

```
490  <!-- Semantic Interoperability Document Definition -->
491  <!-- the Document element holds metadata about the document: -->
492
493  <!ELEMENT Document (Taxonomy+, Assembly, ContextRules?,Component+) >
494
495  <!-- - Taxonomy points to the specific context that, combined with
496  context rules and assembly rules, produced the specific instance.
497  The content of the Taxonomy element is the value or values specified
498  from the referenced context taxonomy.
499  - Assembly references the assembly that produced the instance.
500  - ContextRules references the context rules that produced the
501  instance.-->
502  <!ATTLIST Document
503          name      CDATA #IMPLIED
504          UUID      CDATA #IMPLIED
505  >
506
507  <!ELEMENT Taxonomy (#PCDATA)>
508  <!ATTLIST Taxonomy
509          context   CDATA #REQUIRED
510          ref       CDATA #REQUIRED
511          UUID      CDATA #IMPLIED
512  >
513
514  <!ELEMENT Assembly EMPTY>
515  <!--     For each specified contextual value for the document, you must
516  supply a context name and a value, expressed as the name of the context
517  driver (the top level of the context hierarchy), an equals sign, and
518  one or more values enclosed in single quotes. For example:
519
520          value="Industry='Aerospace' Geopolitical='United States'"
521
522  Note that ranges in the value position are indicated by hyphens and
523  that path expressions are valid values. Lists of values may be
524  indicated by using commas or pipes, with or without whitespace.
525  -->
526  <!ATTLIST Assembly
```

```
527              name      CDATA #REQUIRED
528              value     CDATA #REQUIRED
529              UUID      CDATA #IMPLIED
530  >
531  <!ELEMENT ContextRules EMPTY>
532  <!ATTLIST ContextRules
533              name      CDATA #REQUIRED
534              value     CDATA #REQUIRED
535              UUID      CDATA #IMPLIED>
536
537  <!ELEMENT Component (Component | Group)*>
538
539  <!--    - Type attribute must be included if the element is of a simple
540  type. If it is not provided, the name value is assumed to be the same
541  as the complex type name.
542          - Occurrence applies to the component itself and indicates how
543  often it occurs in the final schema.It must be one of the following:
544                  [no value is "one and only one"]
545                  ?
546                  +
547                  *
548                  n,m where n is minimum and m is maximum
549
550
551  - Sequence applies to the children of the component. It is information
552  in the context rules that must be kept, even if not all syntaxes need
553  it or support it. Values should be:
554                  FollowedBy: the order in which the children are
555  specified is important, and is
556  the order in which they will be specified in the final schema.
557                  AnyOrder: the order in which the children are specified
558  is not important, since the final schema will allow them in any order.
559  All of the children must be present in a document written according to
560  the final schema.
561                  Choice: the order in which the children are specified
562  is not important. Only one of the children is allowed in a document
563  written according to the final schema.
564   -->
565  <!ATTLIST Component
566              name      CDATA #REQUIRED
567              type      CDATA #IMPLIED
568              occurrence CDATA #IMPLIED
569              sequence  CDATA #IMPLIED
570              UUID      CDATA #IMPLIED
571  >
572
573  <!-- The Group element functions as a way of describing the structural
574  relationships among nested, unnamed groups of child components. The use
575  of its attributes are the same as for the Component elements.
576  -->
577  <!ELEMENT Group (Component | Group)*>
578  <!ATTLIST Group
579              occurrence CDATA #IMPLIED
580              sequence  CDATA #IMPLIED
581  >
```

582 **10 Output Constraints**

583 Documents produced through the application of Assembly and Context Rules must
584 contain information regarding which rules and context were used as metadata.

585   # 11 Appendix: Examples

586   ## 11.1 Example of Assembly Rules document

```
587     <?xml version="1.0"?>
588     <!DOCTYPE Assembly SYSTEM "assembly.dtd">
589     <Assembly version="1.0">
590       <Assemble name="PurchaseOrder" id="PO">
591         <CreateGroup>
592           <CreateElement location="UUID" id="Buyer">
593             <Name>Buyer</Name>
594             <Type>PartyType</Type>
595             <CreateGroup>
596               <UseElement name="Name">
597               </UseElement>
598               <UseElement name="Address">
599                 <CreateGroup id="fred">
600                   <CreateGroup type="choice">
601                     <UseElement name="BuildingName">
602                     </UseElement>
603                     <UseElement name="BuildingNumber">
604                     </UseElement>
605                   </CreateGroup>
606                   <UseElement name="StreetName">
607                   </UseElement>
608                   <UseElement name="City">
609                   </UseElement>
610                   <UseElement name="State">
611                   </UseElement>
612                   <UseElement name="ZIP">
613                   </UseElement>
614                   <UseElement name="Country">
615                   </UseElement>
616                 </CreateGroup>
617               </UseElement>
618             </CreateGroup>
619             <Condition test="$Geopolitical='United States'">
620               <Rename from="address" to="addressUS"/>
621               <Rename from="Place" to="City"/>
622               <Rename from="address/County" to="State"/>
623               <Rename from="address/PostalCode" to="ZIP"/>
624             </Condition>
625           </CreateElement>
626           <CreateElement id="Seller" location="UUID">
627             <Name>Seller</Name>
628             <Type>PartyType</Type>
629           </CreateElement>
630         </CreateGroup>
631         <CreateElement location="UUID" id="Item">
632           <Name>Item</Name>
633           <Type>ItemType</Type>
634           <MinOccurs>1</MinOccurs>
635           <MaxOccurs>unbounded</MaxOccurs>
636         </CreateElement>
637       </Assemble>
```

Document Assembly and Context Rules   Page 23 of 30

```
638      <Assemble name="PurchaseOrderReceipt" id="POR">
639        <CreateGroup>
640          <CreateElement idref="Seller">
641          </CreateElement>
642          <CreateElement idref="Buyer">
643          </CreateElement>
644        </CreateGroup>
645        <CreateElement idref="Item">
646        </CreateElement>
647        <CreateElement location="UUID" id="Ack">
648          <Name>Acknowledgment</Name>
649          <Type>AckType></Type>
650        </CreateElement>
651      </Assemble>
652    </Assembly>
```

11.2 Example of Context Rules Document

```
654    <?xml version="1.0"?>
655    <!DOCTYPE ContextRules SYSTEM "contextrules.dtd">
656    <ContextRules id="CalAer">
657      <Rule apply="hierarchical">
658        <Taxonomy context="Geopolitical"
659          ref="http://ebxml.org/classification/ISO3166"/>
660        <Taxonomy context="Industry"
661          ref="http://ebxml.org/classification/industry/aviation"/>
662        <Condition test="$Geopolitical='United States'">
663          <Action applyTo="//Buyer/Address">
664            <Occurs>
665              <Element >
666                <Name>State</Name>
667              </Element>
668            </Occurs>
669            <Add after="@id='fred'">
670              <CreateGroup type="choice">
671                <Element >
672                  <Name>Floor</Name>
673                  <Type>string</Type>
674                </Element>
675                <Element >
676                  <Name>Suite</Name>
677                  <Type>string</Type>
678                </Element>
679              </CreateGroup>
680            </Add>
681            <Condition
682              test="$Geopolitical='California' and$Industry='Aerospace'">
683              <Occurs>
684                <Element >
685                  <Name>ZIP</Name>
686                </Element>
687              </Occurs>
688            </Condition>
689          </Action>
690        </Condition>
691      </Rule>
```

```
692    <Rule order="10">
693      <Taxonomy context="Geopolitical"
694      ref="http://ebxml.org/classification/ISO3166"/>
695      <Condition test="$Business Process='RFQ'">
696        <Condition test="Industry='Insurance'">
697          <Action applyTo="//Party">
698            <Add before="Address">
699              <Element >
700                <Name>QualifyingInfo</Name>
701                <Type>QualifyingInfo</Type>
702                <Attribute>
703                  <Name>Privacy</Name>
704                  <Type>yes | no</Type>
705                  <Use>default</Use>
706                  <Value>yes</Value>
707                </Attribute>
708                <Attribute>
709                  <Name>Accuracy</Name>
710                  <Type>CDATA</Type>
711                  <Use>required</Use>
712                </Attribute>
713                <Annotation>
714                  <Documentation>What this element is for.
715                  </Documentation>
716                </Annotation>
717              </Element>
718            </Add>
719          </Action>
720        </Condition>
721        <Condition test="$Industry='Travel'">
722          <Action applyTo="//Party">
723            <Subtract>
724              <Attribute >
725                <Name>TaxIdentifier</Name>
726              </Attribute>
727            </Subtract>
728          </Action>
729        </Condition>
730      </Condition>
731    </Rule>
732    <Rule>
733      <Taxonomy context="Industry"
734        ref="http://ebxml.org/classification/Industry/Automotive"/>
735      <Condition test="$Industry='Automotive'">
736        <Action applyTo="//QualifyingInfo">
737          <Add>
738            <Element >
739              <Name>DrivingRecord</Name>
740              <Type>DrivingRecord</Type>
741            </Element>
742            <Element >
743              <Name>CarDescription</Name>
744              <Type>CarDescription</Type>
745            </Element>
746            <Element >
747              <Name>DrivingHabits</Name>
```

```
748                <Type>DrivingHabits</Type>
749              </Element>
750            </Add>
751            <Rename from="@Convictions" to="@DrivingConvictions"/>
752          </Action>
753          <Action applyTo="//QualifyingInfo/@Convictions">
754            <Add>
755              <Attribute>
756                <Value>Unknown</Value>
757              </Attribute>
758            </Add>
759          </Action>
760        </Condition>
761      </Rule>
762  </ContextRules>
```

11.3 Example of Semantic Interoperability Document

This example assumes a US address, and the California/Aerospace example from above.

```
<?xml version="1.0"?>
<!DOCTYPE Document SYSTEM "sid.dtd">
<Document>
  <Taxonomy context="Geopolitical"
    ref="http://ebxml.org/classification/ISO3166">Region
  </Taxonomy>
  <Assembly name="PurchaseOrder" value="Geopolitical='United States'"/>
  <ContextRules name="CalAer" value="Industry='Aerospace'
    Geopolitical='United States'"/>
  <Component name="PurchaseOrder" sequence="FollowedBy">
    <Component name="Buyer" sequence="FollowedBy">
      <Component name="Address" sequence="FollowedBy">
      <Group sequence="Choice">
        <Component name="BuildingName"/>
          <Component name="BuildingNumber"/>
        </Group>
        <Group sequence="Choice">
          <Component name="Floor"/>
          <Component name="Suite"/>
        </Group>
        <Component name="City"/>
        <Component name="State"/>
        <Component name="ZIP"/>
        <Component name="Country"/>
      </Component>
    </Component>
    <Component name="Seller"/>
    <Component name="Item" occurrence="+"/>
  </Component>
</Document>
```

797   **12 References**

798   [XPATH]   http://www.w3.org/TR/xpath

799 **13 Disclaimer**

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

804 **14 Contact Information**

805   Team Leader
806   Name   Arofan Gregory
807   Company   Commerce One
808   Street   Vallco Parkway
809   City, state, zip/other   Cupertino, CA
810   Nation   US
811
812   Phone:
813   Email: arofan.gregory@commerceone.com
814
815   Vice Team Lead
816   Name   Eduardo Gutentag
817   Company   SUN Microsystems
818   Street   17 Network Circle – UMPK17-102
819   City, state, zip/other   Menlo Park, California
820   Nation   US
821
822   Phone: +1-650-786-5498
823   Email: Eduardo.Gutentag@eng.sun.com
824
825   Editor
826   Name   Gait Boxman
827   Company   TIE
828   Street   Beech Avenue 161
829   City, state, zip/other   Amsterdam (Schiphol-Rijk)
830   Nation   The Netherlands
831
832   Phone:
833   Email: gait.boxman@tie.nl
834

Copyright Statement

836 Copyright © UN/CEFACT and OASIS, 2001. All Rights Reserved.
837
838 This document and translations of it MAY be copied and furnished to others, and
839 derivative works that comment on or otherwise explain it or assist in its implementation
840 MAY be prepared, copied, published and distributed, in whole or in part, without
841 restriction of any kind, provided that the above copyright notice and this paragraph are
842 included on all such copies and derivative works. However, this document itself MAY
843 not be modified in any way, such as by removing the copyright notice or references to
844 ebXML, UN/CEFACT, or OASIS, except as required to translate it into languages other
845 than English.
846
847 The limited permissions granted above are perpetual and will not be revoked by ebXML
848 or its successors or assigns.
849
850 This document and the information contained herein is provided on an "AS IS" basis and
851 ebXML DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING
852 BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
853 INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
854 WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR
855 PURPOSE.