



Creating A Single Global Electronic Market

1
2
3
4
5
6
7
8
9
10
11
12
13
14

Context and Re-Usability of Core Components

ebXML Core Components

10 May 2001
Version 1.04

15 **1 Status of this Document**

16 This Technical Report document has been approved by the Core Component Project
17 Team and has been accepted by the ebXML Plenary.

18

19 This document contains information to guide in the interpretation or implementation of
20 ebXML concepts.

21

22 Distribution of this document is unlimited.

23

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

25

26 This version:

27 www.ebxml.org/specs/ebCNTXT.pdf

28

29 Latest version:

30 www.ebxml.org/specs/ebCNTXT.pdf

31

32 **2 ebXML participants**

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

35

36 Editing team: Mike Adcock, APACS
37 Sue Probert, Commerce One
38 James Whittle, e CentreUK
39 Gait Boxman, TIE
40 Thomas Becker, SAP

41

42 Team Leader: Arofan Gregory, Commerce One

44

45 Vice Team Leader: Eduardo Gutentag, SUN Microsystems

47

48 Contributors:

49 Tom Warner
50 Jim Dick
51 Rob Jeavons
52 David Connelly
53 Arofan Gregory
54 Martin Bryan
55 Mike Adcock
56 Eduardo Gutentag
57 Matthew Gertner
58 Polly Jan
59 Sharon Kadlec
60 Sally Wang
61 James Wertner
62 Todd Freter
63 Henrik Reiche
64 Chris Nelson
65 Martin Roberts
66 Samantha Rolefes

67	3 Table of Contents	
68	1 Status of this Document	2
69	2 ebXML participants	3
70	3 Table of Contents	4
71	4 Introduction	5
72	4.1 Summary of Contents of Document	5
73	4.2 Context Defined	5
74	4.3 Context in a business perspective	6
75	5 Using Context Descriptors	8
76	5.1 Context-controlled Core Component Metamodel	8
77	5.1.1 Core Component Type Definitions	8
78	5.1.2 Basic Information Entity	8
79	5.1.3 Aggregate Information Entity	9
80	5.1.4 Functional Set	9
81	5.2 Context Constraints	9
82	5.3 Seeding Core Components	10
83	5.4 Using Core Components	10
84	5.5 Building Business Documents	10
85	5.6 Beyond Re-use	11
86	5.7 Non-compliance Issue	11
87	6 The Application of Context to Business Problems	12
88	6.1 Promoting Interoperability	12
89	6.1.1 Using Context to Handle Name and Structural Location Variation When	
90	Determining Semantic Equivalence	12
91	6.1.2 Reusing Data Across Related Processes	13
92	6.1.3 International and Cultural Variation in Data	14
93	6.2 Implementation Strategies for Core Component Context	15
94	6.2.1 Common Core Component Context Implementation Considerations	15
95	6.2.2 Browser-Hosted Implementation Strategy	15
96	6.2.3 ERP and EDI Integration	16
97	7 Disclaimer	18
98	8 Contact Information	19
99	Copyright Statement	20

100 **4 Introduction**

101 **4.1 Summary of Contents of Document**

102 This document describes how business contexts' influence on data structures can be
 103 rendered in an explicit, machine-processable form. This is done by establishing a set of
 104 classification hierarchies that are used to identify the situations in which a core
 105 component will require modification. The classifications that are being recommended are
 106 to be found in ebXML TR - Catalogue of Context Drivers Ver 1.04. The methodology for
 107 the use of these context drivers is detailed in ebXML TR – Document Assembly and
 108 Context Rules Ver 1.04.

109
 110 The present document **MUST** be read in conjunction with these documents. The purpose
 111 of this document is to give readers sufficient familiarity with the idea of explicit
 112 utilization of context drivers to enable them to understand the classifications and
 113 methodology as described in those documents.

114
 115 The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**,
 116 **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL**, when they appear in this
 117 document, are to be interpreted as described in RFC 2119.

118
 119 **4.2 Context Defined**

120 When a business process is taking place, the context in which it is taking place can be
 121 specified by a set of contextual categories and their associated values. For example, if an
 122 glue manufacturer is selling to a shoe manufacturer, the context values might be as
 123 follows:

124

Contextual Category	Value
Process	Procurement
Product Classification	Glue
Region (buyer)	France
Region (seller)	U.S.
Industry (buyer)	Garment
Industry (seller)	Adhesives

125

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

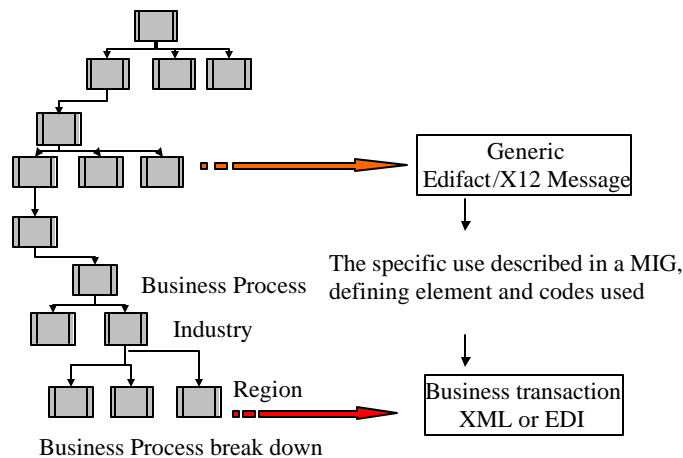
128

- 129 • Design Time - to create the minimum useful schema.
- 130 • Integration Time - Identify and help resolve data requirements conflicts required
 131 for business transactions.
- 132 • Run Time - to express the business relationships between data.

- 133 ○ Used by Trading Partners to validate the runtime document instances.
- 134 • Navigation of the registry to find other data sets.
- 135 ○ Need to hold the data about the context in the rules.
- 136 • Discovery Process for creating Core Components or extensions.
- 137 ○ Core Components are discovered along with the business context in which
- 138 they are used.
- 139
- 140 For a catalogue of Contexts, see ebXML TR - Catalogue of Context Drivers Ver 1.04.

141 **4.3 Context in a business perspective**

142 The concept of context is not new. It can be found already in existing messages like
 143 EDIFACT or X12. Context is one of the aspects of modelling business processes, as
 144 illustrated in the following example, which shows the top-down modelling of a business
 145 process into more and more specific processes:



146 Although UML modelling of business processes is discussed as if it is a completely new
 147 approach, it is not. Earlier development of EDI messages was done by identifying
 148 business processes. Typically the underlying process was defined in some generic way,
 149 describing the specific data elements and codes to be used, while leaving it to
 150 implementations to define the specific use of the message. The Message Implementation
 151 Guides describe a subset of a generic message, where specific elements qualified by
 152 codes express specific data (semantics). The overall term for this expression of specific
 153 data is what we define as *context*.

154
 155
 156 The diagram above also provides an example of where context is used. Breaking down a
 157 business process implies the application of some of the major context drivers.
 158

159 Context for a business process is one-dimensional, and includes two roles in an industry
160 in a region with respect to an official constraint, for instance. These context drivers are
161 not applied in some sequence: they form *the* context for the business process.
162
163 Besides business process context-drivers there may be other activity context-drivers,
164 which again are not applied in some sequence, but form *the* context for the business
165 activity.
166
167 The technical application of the Core Components context drivers requires a
168 methodology for using context to define transactions. The business perspective of context
169 is well known and used by implication today. The rest of this technical report, and the
170 ones related to it (ebXML TR - Document Assembly and Context Rules Ver 1.04 and
171 ebXML TR - Catalogue of Context Drivers Ver 1.04) enable an explicit expression and
172 use of context.

173 **5 Using Context Descriptors**

174 **5.1 Context-controlled Core Component Metamodel**

175 The formal model for the Context-controlled Core Component Metamodel can be seen in
176 the document ebXML TR - Catalogue of Context Drivers Ver 1.04.

177

178 **5.1.1 Core Component Type Definitions**

179 A Core Component Type Definition defines a reusable type of core component for which
180 no pre-determined use name has been assigned. No business semantics are associated
181 with the Core Component Type Definition – these semantics appear when it is used in a
182 Basic Information Entity.

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

185

186 A human-readable name for the type (ending in the word Type, e.g. AmountType), and a
187 brief description of the purpose of the type, are also required. For further specification see
188 the document ebXML TR - CC Dictionary Entry Naming Conventions Ver 1.04.

189

190 By default a Core Component Type Definition is deemed to be restrictable or extendable.
191 If this is not the case the isRestrictable or isExtendable boolean properties must be set to
192 False. This is also true of Basic and Aggregate Information Entities.

193

194 **5.1.2 Basic Information Entity**

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

198

199 A formal definition of the relevant Datatype must be associated with each Basic
200 Information Entity. This could be done in accordance with Part 2 of the W3C's XML
201 Schema specification, or using Document Type Definitions as specified in the W3C
202 XML 1.0 specification.

203

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

207

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

214

215 **5.1.3 Aggregate Information Entity**

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

221

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

224

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

228

229 **5.1.4 Functional Set**

230 A Functional Set is a set of two or more Functional Sets, or two or more Basic
231 Information Entities or Aggregates that can be used to model information related to a
232 single function in different ways.¹

233

234 **5.2 Context Constraints**

235 A Document Model is created by applying a set of Context Rules to a set of Basic and/or
236 Aggregate Information Entities that have been “assembled” to meet a defined business
237 process.

238

239 The Assemble Types modelling element identifies the base Basic and/or Aggregate
240 Information Entities, applies an appropriate sequence to the components and renames
241 embedded components as required within the business process.

242

243 The Context Constraints define modifications to be made to existing Basic and/or
244 Aggregate Information Entities when used within specific contexts, and any Application
245 Component needed to extend a core component or the document model.

246

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

249

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

254

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.

255 **5.3 Seeding Core Components**

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

260
261 Recognizing that there are situations in which equivalent information can be expressed in
262 several ways, relevant core components can be grouped together into Functional Sets.
263 These provide a means by which a limited choice of stereotype information entities can
264 be offered as alternative ways of specifying information for a particular function, e.g. a
265 location can be specified as an address, a GPS reference, or a UN Locode. While the
266 functional set is still a stereotype, the choice is dependent on a business context or
267 contexts.

268

269 **5.4 Using Core Components**

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

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

276

277 Use of a core component with extensions (or indeed restrictions) in a particular business
278 context creates a Process Specific Entity. This is registered under a unique business name
279 formed from the context and the stereotype component names.

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

283

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

288 **5.5 Building Business Documents**

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

294

295 If no appropriate context constrained information entity exists, a new one must be
296 created, according to the principles described in the previous section, and ideally using an

297 existing stereotype. Registration of the new process specific information entity adds to
298 the range of available context-constrained information entities..

299 **5.6 Beyond Re-use**

300 If no appropriate existing stereotype exists, an industry vertical or similar community
301 may need to:

- 302 • Create additional Basic components for pieces of information, which cannot be
303 represented using already-defined Core Components. These are Domain Basic
304 Components.
- 305 • Use Core Component(s) to construct a non-core Aggregate Component, called a
306 Domain Complex Component.
- 307 • Use Core Component(s) and Domain Components to construct a non-core
308 Complex Component, also known as a Domain Complex Component.
- 309 • Use Domain Component(s) to construct a non-core Complex Component. These
310 are also Domain Complex Components.

311

312 Ideally, Domain Components need to be recorded in the same detail as Core
313 Components, complete with relevant Context(s). This is an aspect of extensibility;
314 Domain Components should be registered so as to avoid 're-inventing the wheel'.
315 Newcomers can re-use Domain Components and register any additional Context(s) with
316 which they will henceforth be associated

317 At some point, non-core Domain Components can become Core Components, according
318 to criteria that judge the degree of re-use. These values would be the same as values
319 found in a business process model that informs the contextual use of the core
320 components.

321

322 **5.7 Non-compliance Issue**

323 This section raises two basic issues:

- 324 1) Extensibility
- 325 2) Registration

326

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

330

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

334

335 Unregistered Domain Components:

- 336 • Will hinder communication and interoperability between different communities.
- 337 • Should not, in any circumstances, be favoured over formally registered
338 equivalents.

339 **6 The Application of Context to Business Problems**

340 This section offers a discussion of how context can be deployed to solve real-world
341 problems of interoperability and document design. It makes no claims to being
342 comprehensive.
343

344 **6.1 Promoting Interoperability**

345 A few of the common scenarios faced by trading partners today are:

- 346 • **Same Data, Different Names:** Frequently, trading partners are asked to support
347 multiple sets of business vocabularies, where the same data is referred to with two
348 or more different names. Typically, the equivalence is established using mapping
349 and translation tools and code conversions, requiring extensive work to integrate
350 systems.
- 351 • **Same Data, Different Structural Position:** This is a related problem - the same
352 piece of data may be located in different places structurally in equivalent
353 messages.
- 354 • **Same Data, Different Process:** Because of differences in business process, the
355 same data may be expressed differently. Often, this is seen when the same basic
356 message structure is used in two related processes, but the cardinality of some
357 data members is different based on where the message is being used.
- 358 • **Same Data, Different Culture:** This is a case most often seen in international
359 trade, where different cultures format and structure data differently from other
360 cultures.

361

362 For each of these scenarios, we will look at how the application of context can promote
363 interoperability. In each case, it is assumed that the trading partners describe the data
364 needs for each business process they support in the form of Assembly and Context rules.
365 These can then be made available in a repository, or be given directly to prospective
366 trading partners. Specific implementation options are discussed in more detail below.
367 Please note that all examples given are meant to be illustrative, and may not be based
368 very firmly in reality.

369

370 **6.1.1 Using Context to Handle Name and Structural Location Variation** 371 **When Determining Semantic Equivalence**

372 This section addresses the first two scenarios listed above. One place where this type of
373 lack of interoperability is seen is in supply chain scenarios, where small suppliers are
374 selling into more than one industry vertical.
375

376

377 Industry "verticals" are generally defined by the large buyers at the top of the supply
378 chain. Large buyers have highly automated back-office systems; smaller suppliers do not.
379 Because "industries" view things from their own perspective, they tend to organize data
380 differently, and they often use taxonomies that are specific to their industry. Conversely,
smaller suppliers often produce goods and services for many different industries: a glue

381 manufacturer could sell a product used in making planes, cars, and shoes, for example,
382 which are seen as three completely separate industry verticals.
383 Since each industry vertical has different names for the same things, and arranges data
384 differently, it is difficult or impossible for SMEs to fully automate their business. The
385 time for data to travel up and down the supply chain is therefore very long, inventories
386 must be kept high, and many potential efficiencies are lost all throughout the supply
387 chain.

388

389 For example, when our small glue manufacturer receives orders from two of these
390 industries, they will have different "standard" vocabularies. Let's say that in the
391 automotive vocabulary, the requested date for shipping each item in an order is called
392 `ShippingDate`, and that this information is always included with each item in the
393 order. For the clothing manufacturer, the same information is a `ShipDate` and it is
394 located only once in the header.

395

396 Today, this kind of problem would be handled by translation. A transformation tool
397 would map between these obviously corresponding pieces of data. By analyzing the
398 various vocabularies that must be supported, the glue manufacturer would be able to
399 create a map for each industry standard or trading partner vocabulary supported. The
400 problem here is basically one of cost: an expensive analysis must be conducted to
401 determine the equivalencies in each vocabulary, even when they are fairly obvious.

402

403 The automation of this mapping process is enabled by Semantic Identification
404 Documents, which describe a document's derivation from Assembly and Context Rules,
405 and Assembly and Context rules, which describe the derivation of each industry's
406 vocabulary from a set of core components. In each case, the semantics of the data can be
407 identified by tracing them back to the core component from which they were derived.

408

409 Because the core component that exists as the basis of any vocabulary can be traced back
410 through this chain, the base semantic of any field or message structure can be determined.
411 By mapping each piece of data in each document structure back to its core, and then
412 comparing the two, equivalence can be automatically determined, and a mapping derived
413 for use by a transformation engine. Note that this process may also require a knowledge
414 of the parent-child relationships between components, as these provide semantic
415 qualification of the core. (For example, a `Tax` element inside a line item has potentially
416 different semantic relevance than the same component used at the header level.)

417

418 Ultimately, the cost of developing the mapping for translation technology is reduced,
419 because the extensive manual analysis formerly required is no longer needed. While this
420 does not entirely remove the cost of integrating a new trading partner, it does provide a
421 significant reduction in cost.

422

423 ***6.1.2 Reusing Data Across Related Processes***

424 Very often, a single item of data is used in multiple transactions within a single business
425 process, or is used in two related business processes. In many cases, a single message
426 structure can be used to support these different processes or related transactions. An

427 example of this includes an Order, which may be used to request a purchase order
428 (OrderRequest), to place an order (Order), and to do change ordering
429 (ChangeOrder). These three transactions require a nearly identical set of data, but are
430 different. Typically, these differences stem from some action or status related to a
431 specific point in the business process, or involve the ability of one trading partner to
432 include data that may not yet be available at the time a message is created.

433

434 In a description of this document structure, fields must be provided for all of the data
435 required at every stage of the process. At the same time, anything that cannot be included
436 at every point across the business process must be made optional. (This is less of an issue
437 with EDI syntax, since all that needs to be changed is the implementation guide that
438 discusses the use of the document. In XML, either an entirely new document type must
439 be described, or a field must be made "optional" that might be better "required" at some
440 other point in the process.)

441

442 In order to achieve tight validation, a separate document description for each transaction
443 must be available. If what is wanted is the simplicity offered by having a single document
444 type, then validation must be sacrificed (particularly for XML systems). This is a
445 problem that can be solved through the application of context.

446

447 By specifying the needed data and optionality *within a single document type* through
448 context rules, and tying these to a specific transaction or point within the business
449 process, the advantage of smaller, more specific documents, and a single base document
450 type can be achieved. The process described above for tracing a data element back
451 through the Context Rules and Assembly Rules to a specific core component is used
452 again here, although this is typically a design-time activity that does not need to be
453 performed by an application.

454

455 ***6.1.3 International and Cultural Variation in Data***

456 It is very often the case that a single set of business data is structured differently in
457 different parts of the world. Often, this is a reflection of cultural differences in the real
458 world. Perhaps the best-known example of this is the structuring of addresses, which
459 reveal a huge amount of variation. It is certainly possible to store all potentially useful
460 address-related information in a back-office system, but, depending on where the trading
461 partners are and what their data demands are, they will probably only be capable of
462 processing a small number of the possible structural variations.

463

464 Context provides a clear way of dealing with this situation: every trading partner can
465 fully describe their structural needs in Context Rules, and the semantic equivalency of
466 different fields can be established using the mechanism described above. This allows us
467 to determine the correct structures for each trading partner, based on where they do
468 business.

469

470 **6.2 Implementation Strategies for Core Component Context**

471 Different use cases will require different implementation strategies for taking advantage
472 of core component context. In the case of smaller companies with minimal back-office
473 software in place, a browser-hosted solution using web forms for data entry may be the
474 best choice for integrating with trading partners. Larger companies will need more
475 sophisticated solutions that bind into ERP systems on the back-end, and provide
476 connectivity with EDI gateways for integration with trading partners who have not
477 implemented ebXML. In both cases, it must be possible to perform integration both at
478 design-time and at run-time. Design-time integration is likely to be the standard case,
479 especially in the short term, but run-time integration will yield the most value over the
480 long-term, since it will enable on-the-fly discovery of new trading partners and
481 negotiation of mutually acceptably data forms, without the need for expensive and time-
482 consuming manual integration work.

483

484 **6.2.1 Common Core Component Context Implementation Considerations**

485 In all integration scenarios, the same underlying process is engaged in order to implement
486 core component context. A context engine is fed the appropriate assembly and context
487 rules for both trading partners, identifying the core components that make up the business
488 documents for a given business process and any modifications that must be made to these
489 core components in order to meet specific trading partner requirements.

490

491 The assembly rules are applied first, resulting in a schema or DTD modeling the relevant
492 information. (For the sake of simplicity, we will use the term “schema” in subsequent
493 discussion to refer to any one of the various dialects of XML schemas and to DTDs.)

494 Context rules are then applied to adapt the schema to the contexts in which the trading
495 partners are active. The output is thus a customized schema that contains all of the
496 necessary information for the interaction, using standard core components wherever
497 possible to maximize interoperability.

498

499 In order to achieve run-time integration, additional information, known as schema
500 annotations, must be made available at the core component level to specify bindings to
501 ERP systems, EDI gateways and web forms. These annotations reference standard core
502 components, once again for interoperability purposes. The annotations, on the other hand,
503 are trading-partner-specific and, in essence, tell the run-time integration engine how to
504 marry these standard core components with the implementation details of the systems
505 used by each company.

506

507 **6.2.2 Browser-Hosted Implementation Strategy**

508 Small companies that do not have back-office software in place conduct business
509 primarily using phone and fax. For them, manual data processing is an integral part of
510 trading partner integration. Significant value can be gained from use of core components
511 by replacing these existing systems with browser-hosted applications that go straight
512 from a web form to an ebXML-conformant XML document that can be transmitted
513 directly to a trading partner. Conversely, incoming data in the form of XML messages
514 can be displayed in the browser.

515

516 If a company wishes to perform design-time integration with a specific trading partner, a
517 schema is first generated that takes into account the requirements of the two parties
518 (using the context engine described above). Two primary interfaces must then be
519 implemented based on the data model described by this schema. The first interface
520 enables the company to view incoming XML documents. This can be achieved by simply
521 applying an XSLT stylesheet to the document to generate an HTML document that can
522 be shown in a browser. The second interface is more complex, and must enable the
523 company to enter data that will be used to create a schema-conformant XML document
524 that will be communicated to the trading partner. This form can be developed using any
525 standard web development technology.

526

527 The main advantage of design-time integration is that it does not require any special
528 technology other than what is commonly available today. On the other hand, the manual
529 development of the kinds of sophisticated web forms needed for real-world
530 implementations of complex schemas is quite challenging and time-consuming. The use
531 of tools that automate this process by generating forms directly from schemas can be
532 highly advantageous, to the extent that these tools are available.

533

534 In the case of run-time integration, even consultation of incoming documents is more
535 complex than in the design-time scenario. Since the schema is not known ahead of time,
536 so it is not possible to author an XSLT stylesheet to do an XML to HTML mapping. One
537 solution would be to display the documents as raw XML using XML display capabilities
538 such as those included in Internet Explorer 5.0. This is not entirely satisfying, however,
539 as the raw XML view is neither particularly attractive nor intuitive. Otherwise, schema
540 annotations of the type described above can be used to automate the formatting of the
541 document, without the need for a hard-coded stylesheet.

542

543 Creation and modification of outgoing documents at run-time clearly requires the use of
544 some sort of tool capable of generating web forms dynamically from schemas. To a large
545 extent, all of the information necessary for this task is contained in the schema itself:
546 structural information, data types, optionality, etc. Additional information such as field
547 labels, length and ordering can be specified using schema annotations. If XML
548 conformant with the input schema is generated when the form is submitted, the result is a
549 full-fledged system for manual interaction in the web browser with ebXML-compliant
550 systems.

551

552 ***6.2.3 ERP and EDI Integration***

553 For design-time integration with ERP systems and EDI gateways, the schema generated
554 from the assembly and context rules document is used as the basis for the mapping. One
555 option is to write custom integration code that reads the data from the appropriate system
556 (e.g. BAPI calls to retrieve data from an SAP R/3 database) and generates an XML
557 document that conforms to the schema. This is a fairly straightforward process that can
558 leverage a large body of XML processing software.

559

560 Another option is to use one of the increasing number of XML-savvy integration tools.
561 Tools exist already for reading data from a wide range of ERP systems and generating
562 XML documents, and vendors are now announcing support for XML schemas that will
563 partially automate these mappings by reading the schema describing the desired XML
564 document format. The same applies to EDI support; EDI-to-XML gateways exist and are
565 beginning to provide XML schema support that will render the integration task more
566 straightforward.

567

568 When run-time integration is a requirement, the same issue arises as with browser-based
569 integration. The schema is not known ahead of time, so it is not possible to write custom
570 code in order to generate XML documents of the appropriate format. The aforementioned
571 schema-aware integration tools for ERP and EDI represent one possible solution to this
572 problem, to the extent that they are capable of fully automating the binding of schemas.
573 As the schema support provided by these systems matures, it is likely that schema
574 annotations of the type described above will also be used to determine which data in the
575 EDI documents or ERP databases corresponds to which data in the generated XML
576 documents.

577

578 Clearly integration must work in both directions; i.e. it must be possible to read data from
579 an ERP system, and to write data from an incoming XML document back to the ERP
580 system. In the case of EDI systems it will be necessary to convert from EDI to XML and
581 vice versa. While these cases are not always entirely equivalent (e.g. writing back to an
582 ERP system requires concurrency control that is irrelevant when reading from the
583 system), the differences are implementation details that do not change the overall
584 integration strategy.

585 7 Disclaimer

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

590 8 Contact Information

591 Team Leader

592 Name Arofan Gregory
593 Company Commerce One
594 Street Vallco Parkway
595 City, state, zip/other Cupertino, CA
596 Nation US
597
598 Phone:
599 Email: arofan.gregory@commerceone.com
600

601 Vice Team Lead

602 Name Mike Adcock
603 Company APACS
604 Street Mercury House, Triton Court, 14 Finsbury Square
605 City, state, zip/other London EC2A 1LQ
606 Nation UK
607
608 Phone: +44-20-7711-6318
609 Email: mike.adcock@apacs.org.uk
610

611 Editor

612 Name James Whittle
613 Company e centre^{UK}
614 Street 10, Maltravers Street
615 City, state, zip/other London WC2R 3BX
616 Nation UK
617
618 Phone: +44-20-7655-9022
619 Email: james.whittle@e-centre.org.uk

620 Copyright Statement

621 Copyright © UN/CEFACT and OASIS, 2001. All Rights Reserved.

622

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

631

632 The limited permissions granted above are perpetual and will not be revoked by ebXML
633 or its successors or assigns.

634

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